Proyecto LATIn

INICIATIVA LATINOAMERICANA DE LIBROS DE TEXTO ABIERTOS

ALFA
FINANCED BY THE EUROPEAN UNION

Tópicos avanzados de Bases de datos



AUTORES

Cristina Marta Bender
Claudia Deco
Juan Sebastián González Sanabria
María Hallo
Julio César Ponce Gallegos

Tópicos Avanzados de Bases de Datos
1a ed. - Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn), 2014. 115 pag.

Primera Edición: Marzo 2014
Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn)
http://www.proyectolatin.org/

Este texto forma parte de la Iniciativa Latinoamericana de Libros de Texto abiertos (LATIn), proyecto financiado por la Unión Europea en el marco de su Programa ALFA III EuropeAid.

El Proyecto LATIn está conformado por: Escuela Superior Politécnica del Litoral, Ecuador (ESPOL); Universidad Autónoma de Aguascalientes, México (UAA), Universidad Católica de San Pablo, Perú (UCSP); Universidade Presbiteriana Mackenzie, Brasil(UPM); Universidad de la República, Uruguay (UdelaR); Universidad Nacional de Rosario, Argentina(UR); Universidad Central de Venezuela, Venezuela (UCV), Universidad Austral de Chile, Chile (UACH), Universidad del Cauca, Colombia (UNICAUCA), Katholieke Universiteit Leuven, Bélgica (KUL), Universidad de Alcalá, España (UAH), Université Paul Sabatier, Francia (UPS).

Índice general

Introducción

El objetivo de este libro es presentar a los estudiantes una introducción a las nuevas tendencias en bases de datos soportadas por la evolución de la tecnología informática, que contemplan la gestión de nuevos tipos de datos.

La propuesta está dirigida a segundas asignaturas de Bases de Datos y cubre temas tales como la gestión de bases de datos no SQL, datos temporales, datos espaciales, datos en la web, la recuperación de información documental, la creación de repositorios para ayudar a la toma de decisiones, incluyendo una introducción a técnicas y herramientas para analizar estos datos.

Los capítulos "Sistemas de Recuperación de Información", "Bases de Datos y Web", , "Datos Temporales y Datos Espaciales", y "Sistemas de ayuda a la toma de decisión", son aportes de Cristina Bender y Claudia Deco. Juan Sebastián González Sanabria es autor del capítulo "Diseño de Bases de Datos". El capítulo "Bases de Datos NoSQL" fue escrito por María Hallo. El capítulo "Minería de Datos" es una contribución de Julio César Ponce Gallegos.

Breve CV de los autores

Cristina Marta Bender
Universidad Nacional de Rosario, Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Católica Argentina, Campus Rosario, Facultad de Química e Ingeniería
Argentina

Magister en Informática por la Universidad de la República, Uruguay; Ingeniera Electrónica por la Universidad Nacional de Rosario, Argentina. Profesora e investigadora en el Departamento de Sistemas e Informática de la Facultad de Ciencias Exactas, Ingeniería y Agrimensura, Universidad Nacional de Rosario; y en la Facultad Católica de Química e Ingeniería, Campus Rosario de la Universidad Católica Argentina. Participa en proyectos nacionales e internacionales de investigación vinculados a sus intereses de investigación. También es coautora de varios artículos en journals y proceedings de conferencias internacionales y nacionales. Además, es revisora de artículos para diversos congresos. En su actividad docente es tutora y evaluadora de tesinas de grado y de posgrado. Sus intereses de investigación incluyen tecnologías de bases de datos y de recuperación de información (information retrieval).

Claudia Deco
Universidad Nacional de Rosario, Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Católica Argentina, Campus Rosario, Facultad de Química e Ingeniería
Argentina

Doctora en Ingeniería por la Universidad Nacional de Rosario, Argentina. Magister en Informáti-

ca por la Universidad de la República, Uruguay; Licenciada en Matemática por la Universidad Nacional de Rosario, Argentina. Profesora e investigadora en el Departamento de Sistemas e Informática de la Facultad de Ciencias Exactas, Ingeniería y Agrimensura, Universidad Nacional de Rosario; y en la Facultad Católica de Química e Ingeniería, Campus Rosario de la Universidad Católica Argentina. Participa en proyectos nacionales e internacionales de investigación vinculados a sus intereses de investigación. También es coautora de artículos en journals y proceedings de conferencias internacionales y nacionales. Además, es revisora de artículos para diversos congresos. En su actividad docente es tutora y evaluadora de tesinas de grado y de posgrado. Sus intereses de investigación incluyen tecnologías de bases de datos y de recuperación de información (information retrieval)

Juan Sebastián González Sanabria
Universidad Pedagógica y Tecnológica de Colombia
Colombia

Especialista en Bases de Datos, Universidad Pedagógica y Tecnológica de Colombia; Ingeniero de Sistemas y Computaci ón, Universidad Pedagógica y Tecnológica de Colombia. Profesor de la escuela de Ingeniería de Sistemas y Computación de la Universidad Pedagógica y Tecnológica de Colombia e Investigador adscrito al Grupo de Investigación en el Manejo de la Información adscrito a la Direcci ón de investigaciones de la Universidad Pedagógica y Tecnológica de Colombia. Participante activo de proyectos nacionales de investigación vinculados a sus intereses de investigación. Coautor de artículos publicados a nivel nacional e internacional. Dentro de las actividades docentes dirige trabajos de grado de pregrado y orienta asignaturas afines al área de bases de datos.

Maria Hallo
Escuela Politécnica Nacional
Quito-Ecuador

Máster en Informática por la Universidad Notre Dame de la Paix, Bélgica. Ingeniera Quimica por la Escuela Politécnica Nacional, Quito, Ecuador. Sus intereses científicos son Web Semántica, Recuperación de Información, Inteligencia de negocios, Sistemas de Información Geográfica, Preservación Digital. Ha sido Directora de numerosos proyectos en sus temas de interés. Actualmente realiza investigación en conjunto con Universidad de Valladolid (España) en el tema: Tecnologías de Linked Data y su aplicación en administración de versiones de documentos legislativos. Es Profesora en la Facultad de Ingeniería de Sistemas de la Escuela Politécnica Nacional de Quito Ecuador habiendo dictado clases de Diseño de páginas web, Programación con Java, Sistemas de Información Geográfica, Bases de Datos, Inteligencia de Negocios, Auditoría Informática, Planificación Informática. Además ha sido Directora de más de 20 tesis de pregrado en Ingeniería de Sistemas. Ha realizado publicaciones en revistas y congresos nacionales e internacionales.

Julio Cesar Ponce Gallegos
Universidad Autónoma de Aguascalientes, Departamento de Ciencias de la Computación.
México

Doctor en Ciencias de la Computación por parte de la Universidad Autónoma de Aguascalientes (UAA) en el año 2010. Profesor de Tiempo Completo en el Departamento de Ciencias de la

Computación en la UAA, en materias de Licenciatura, Maestría y Doctorado en las áreas de Programación, Inteligencia Artificial y Fundamentos y Teorías Computacionales, Miembro de diversos comités de diseño curricular para programas educativos de Licenciaturas y Posgrado. Responsable de proyectos de Investigación de la UAA y PROMEP. Miembro titular del cuerpo académico CONSOLIDADO (PROMEP): “Sistemas Inteligentes”. Áreas de Investigación en las que ha publicado: Inteligencia Artificial, Minería de Datos, Ingeniería de Software, Uso de las TICs en la Educación. Cuenta con más de 50 publicaciones en congresos y revistas a nacional e internacional, 11 Capítulos de libros y 2 Libros publicados en editoriales e instituciones como IGI-Global, InTech, Nova Publisher, Textos Universitarios (UAA), SMIA.

1 — Diseño de Bases de Datos

Elaborado por: Esp. Juan Sebastián González Sanabria
Grupo de Investigación en el Manejo de la Información
Universidad Pedagógica y Tecnológica de Colombia
Tunja (Boyacá) – Colombia

1.1 Introducción

Con la aparición de nuevas tecnologías para la gestión de bases de datos, en algunas ocasiones se ha dejado de lado el diseño y modelado de las bases de datos relacionales, las cuales, a hoy, aún siguen siendo las de mayor uso. Es por lo anterior que el capítulo se enfoca en presentar desde un punto de vista simple las consideraciones a tener en cuenta en el ciclo de vida de desarrollo de bases de datos relacionales, aplicando estándares y normas de calidad aprobadas por diferentes entes expertos en la temática como la Organización Internacional de Estándares ISO (por las siglas en ingles de International Organization for Standardization).

El ciclo de vida de las bases de datos va desde el proceso de analizar la problemática hasta la implementación de la solución planteada, pero para el presente capitulo se limitará al llamado esquema o modelo entidad relación, en su fase conceptual, lógica y física, sin dejar de lado otros aspectos necesarios para el diseño.

Para el adecuado entendimiento del tema, se iniciará con el Modelo ER y los elementos que lo componen, continuando con el ciclo de vida de las bases de datos, y finalizando con algunos estándares para el diseño.

1.2 Modelado Entidad-Relación (ER)

El modelado Entidad – Relación se plantea por primera vez en artículos publicados por Peter P. Chen en la década de los setenta, donde inicialmente se divulgó como Modelo Entidad – Interrelación, pero posteriormente a traducciones adecuadas y hasta el día de hoy se le conoce como Modelo Entidad – Relación.

Posterior a la década en mención son diversos los autores y expertos que han hecho aportes y cambios al modelo planteado por Chen, motivo en el que radica la dificultad de que no exista a nivel internacional una notación estándar para este modelado. Sin embargo Richard Barker, junto a otros expertos como Ian Palmer, Harry Ellis, propusieron alrededor del año 1981 algunos aportes y cambios que fueron bien recibidos, haciendo que la notación Barker, en honor a su principal expositor, para modelado Entidad – Relación sea la más utilizada en el contexto internacional por sus diferentes beneficios en la interpretación y “lectura” del Modelo. Es de

aclarar que en algunos países la notación Barker también es conocida como la notación "Pata de gallina", por la traducción latina de "crows foot".

Ahora bien, ¿Por qué hacer uso del modelo ER?, la respuesta a este interrogante es simple: El modelo permite una robusta representación de la totalidad la información que se requiere para la base de datos, eliminando redundancias y siendo el punto de partida para socializar con el cliente o usuario final, pues el modelo es de fácil interpretación por parte de este, al presentar una visión simplificada del negocio.

1.2.1 Elementos del Modelo ER

El Modelo ER cuenta con diferentes elementos, estos difieren mayormente en su forma de representación de acuerdo a la notación seleccionada, para el caso cabe recalcar que se utilizará la Notación Barker. Los elementos más significativos son:

- Entidades
- Atributos
- Relaciones

1.2.1.1. Entidades

Una Entidad se puede definir "algo" de importancia dentro del sistema a desarrollar de lo que interesa almacenar todos los datos. Generalmente se suelen definir objetos (carros, personas, entre otros) y eventos (compras, ventas); aunque se debe aclarar que muchos de los eventos, suelen representarse dentro del sistema con objetos (por ejemplo, una venta se representa mediante una factura).

La representación de las entidades se realiza mediante un rectángulo con sus puntas ovaladas, también conocido como rectángulo suavizado. Cada entidad debe poseer un nombre único dentro del modelo y darse en mayúscula, como se ilustra en la figura 1.1.

Figura 1.1: Representación de una entidad



Es de anotar que el nombre de la entidad para el diseño conceptual se suele representar en singular, y para los siguientes (lógica y física) en plural.

Adicionalmente pese a que no existe un tamaño estándar de las entidades, por buenas prácticas de diseño y para facilidad de entendimiento se recomienda que todas las entidades del modelo conserven proporciones similares.

Las Entidades se deben diferenciar de las INSTANCIAS, las cuales corresponden a los valores que puede tener un atributo, en el Cuadro 1.1 se presentan algunos ejemplos de atributos con sus posibles instancias.

Cuadro 1.1: Ejemplos de atributos con sus posibles instancias

Entidad	Ejemplos Instancias
Persona	Ándres Niño, Diego Sullivan M.
Producto	Computador, Televisor
Cargo	Gerente, Diseñador, Analista

En resumen se puede definir que una entidad representa un conjunto de instancias con un interés común dentro de la lógica del sistema.

Existen dos tipos fundamentales de entidades:

- Entidades Fuertes
- Entidades Débiles

Entidades Fuertes:

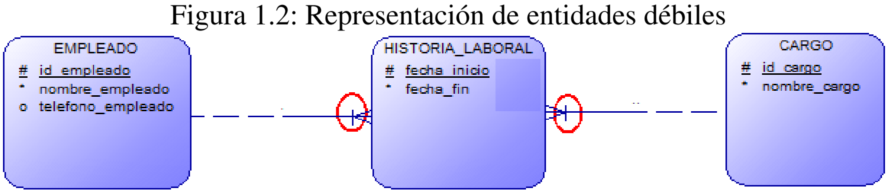
En ocasiones también llamadas entidades regulares, son aquellas que se identifican por sí mismas, es decir, no requieren de otra entidad para existir.

Entidades Débiles:

Para existir dentro del modelo depende de otras entidades. Por ejemplo una entidad HISTORIA_LABORAL solo podría existir si se cuenta con la entidad EMPLEADO y la entidad CARGO. Una entidad débil puede depender de una o muchas entidades.

La representación de las entidades débiles en el modelo se hace con una línea adjunta a la relación, como se ilustra en la Figura 1.2 resaltado en los círculos en rojo:

Figura 1.2: Representación de entidades débiles



Estas entidades débiles suelen aparecer cuando hay atributos asociados a las relaciones.

1.2.1.2. Atributos

Los Atributos, dentro del contexto del Modelo ER, son conocidos como las características de la entidad, aquello que le da propiedades (por ejemplo, el primero nombre, el primer apellido, la fecha de nacimiento de una Persona); los atributos se usan para describir, cuantificar, calificar o clasificar una entidad.

Los atributos deben ser consecuentes con el principio de Atomicidad de las bases de datos, es decir deben tener un único valor. Adicionalmente todo atributo debe corresponder o asociarse a un tipo de dato (texto, numero, fecha, entre otros).

Para representar los atributos en el Modelo ER se escribe el nombre del atributo (singular) dentro de en la entidad en letra minúscula, en caso de que el nombre del atributo sea compuesto por dos o más palabras, el espacio se remplaza por guion bajo (_),

A continuación se presentan ejemplos de algunos atributos asociados a entidades.

Entidad	Atributos
PERSONA	nombre_persona, apellido_persona, fecha_nacimiento
PRODUCTO	nombre_producto, precio_producto
TRABAJO	nombre_profesion, salario_minimo, salario_maximo

Igual que en el caso de las entidades, también se deben diferenciar INSTANCIAS de ATRIBUTOS, por ejemplo:

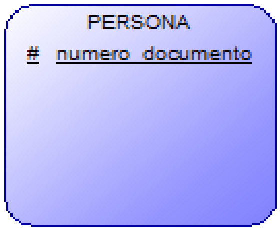
Atributo	Ejemplo Instancia
nombre_persona	Jota
nombre_cargo	Ingeniero
fecha_nacimiento	15/08/1994

Los atributos se clasifican en tres tipos:

- Llave Primaria
- Atributo Obligatorio
- Atributo Opcional.

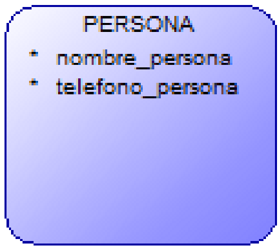
Llave primaria:

Las llaves primarias son aquellas que le brindan una identidad dentro del modelo a la entidad, es aquel atributo cuya INSTANCIA (Valor que puede tomar un atributo) no se va a repetir en ningún caso, como lo puede ser el número de identificación nacional de una persona, el número de matrícula de un carro. Las llaves primarias se representan con el símbolo numeral (#) al lado del nombre del atributo.



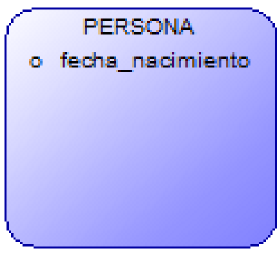
Atributo Obligatorio:

Los atributos obligatorios corresponden a todos los datos que se requieren conocer dentro del contexto del sistema, como puede ser el nombre, el apellido, la dirección y el teléfono de una persona en el caso de un formulario de contacto. Los atributos obligatorios se representan con el símbolo asterisco (*) al lado del



Atributo Opcional:

Corresponden a aquellos datos que podrían llegar a requerirse a futuro o que pueden no tener siempre un valor de instancia requerido, es decir el almacenamiento de estos atributos no influye en el funcionamiento del sistema. Los atributos opcionales se representan con un círculo (o) al lado del nombre del atributo.



TIP 1: Aunque algunos autores sugieren que para el caso de las llaves primarias, al ser también atributos obligatorios, deben representarse con los dos símbolos antes del nombre del atributo; a término de experiencia se sobre entiende que al ser llave es un atributo obligatorio por lo que basta con representarse con el símbolo de numeral.

TIP 2: Si un atributo es opcional u obligatorio, se define por el contexto del sistema que se desarrollará, por tanto en un sistema el atributo *teléfono* puede ser opcional y en otro obligatorio.

TIP 3: Los atributos de un sistema pueden ser considerados como entidades en otro sistema, y viceversa; lo anterior de acuerdo al CONTEXTO en el que se desarrolle.

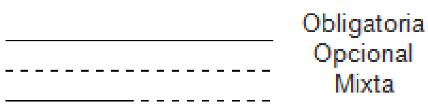
TIP 4: Se debe tener especial cuidado con los campos que son calculados a partir de otros campos, estos son conocidos como CAMPOS VIRTUALES y no se sugiere almacenarlos o representarlos en el modelo.

1.2.1.3. Relaciones

Las relaciones, por redundante que pueda escribirse, representan el cómo las entidades se relacionan entre sí, es decir los vínculos que existen entre los diferentes objetos y eventos que conforman el negocio. Se representan en el modelo mediante líneas que se componen de dos partes fundamentales el grado y la opcionalidad.

Opcionalidad:

Hace referencia a si la relación puede o debe existir, si la relación debe existir se representa mediante línea continua, si la relación puede existir se representa con una línea punteada.

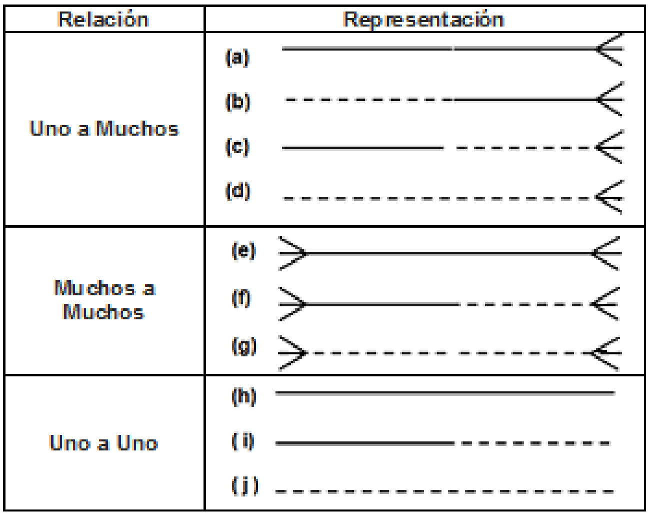


Grado:

Se refiere al número de veces que se puede llegar a relacionar una entidad con otra. Es de su representación de donde se conoce como la notación para Modelado ER de “Pata de gallina”, los grados a representarse son los siguientes:

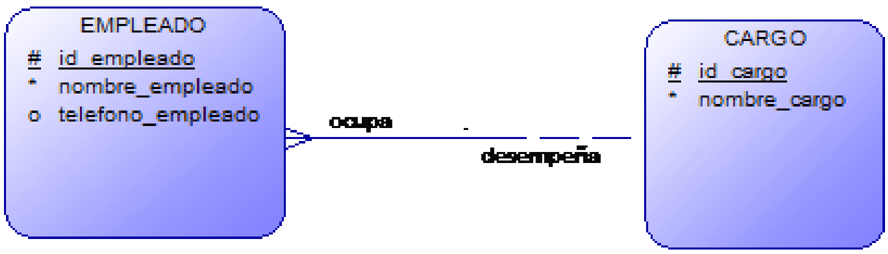


Posterior a estos dos elementos las diferentes combinaciones permiten diez posibles relaciones entre entidades.



A las relaciones se le suelen dar dos perspectivas (nombres de la relación), esto con el fin de dar una adecuada lectura a las relaciones y el modelo sea lo más entendible posible.

La lectura de las relaciones se realiza en el siguiente orden Entidad – Opcionalidad – Grado – Entidad, en el siguiente ejemplo se explica de mejor manera:



De izquierda a derecha:

Entidad	Opcionalidad	Grado	Entidad
Empleado	Debe	Uno	Cargo

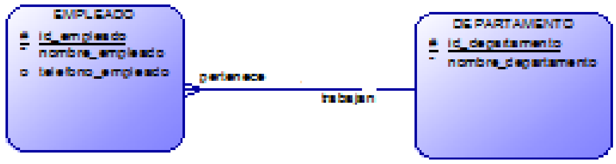
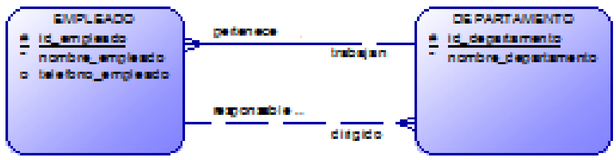
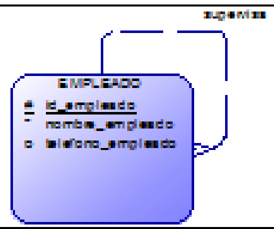
Lo que en lectura vendría dado por *Todo EMPLEADO DEBE ocupar UN CARGOS*.

De derecha a izquierda:

Entidad	Opcionalidad	Grado	Entidad
Cargo	Puede	Muchos	Empleado

Lo que en lectura vendría dado por *Un CARGO PUEDE ser desempeñado por NINGÚN O MUCHOS EMPLEADOS*.

Adicional a las posibles relaciones anteriormente descritas, existen algunos tipos de relaciones de acuerdo a la forma como se establecen entre una o varias entidades.

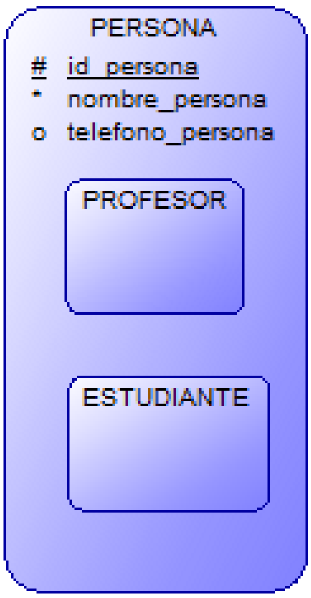
Relación	Descripción	Ejemplo
Binarias	Son las presentadas entre dos entidades.	
Dobles	Son aquellas relaciones que sirven para asociar las mismas entidades, es decir dos relaciones entre las mismas entidades. Estas existen siempre y cuando expresen aspectos diferentes.	
Recursiva	Corresponden a aquellas relaciones que una entidad consigo misma. Generalmente estas relaciones suelen tener una sola perspectiva.	

1.2.1.4. Otros Elementos

Existen una serie de elementos que han surgido para complementar el modelo ER para una mayor interpretación y aproximación a la realidad del sistema. Entre estos elementos cabe destacar los siguientes:

Subtipos:

Los subtipos, también conocidos como subentidades, se incluyen dentro del modelo cuando existen entidades que tienen propiedades en común y que comparten atributos; por ejemplo, las entidades Profesor y Estudiante, pueden ser subtipos de una entidad Persona. Estos se representan con Entidades dentro de una Entidad, esta última conocida como Superentidad.



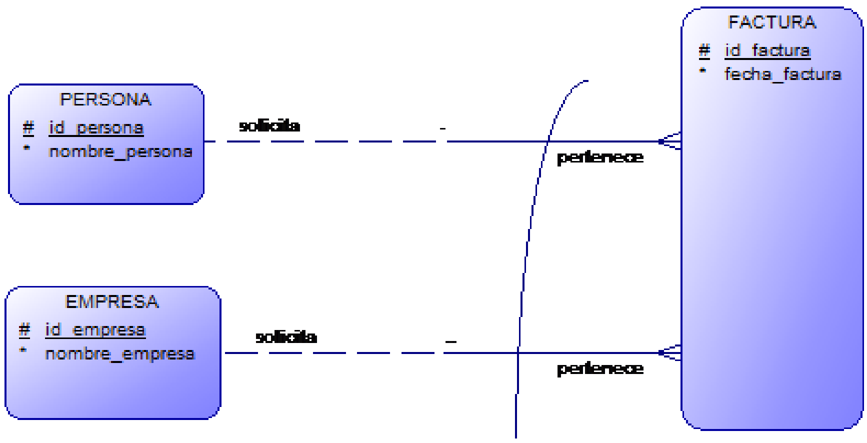
Cuando se hace uso de subtipos es necesario contemplar las siguientes recomendaciones:

- Nunca tienen llaves primarias definidas pues heredan la llave primaria de la entidad que los contiene.
- Cada subtipo puede tener tanto atributos como relaciones propias.
- No existen entidades con un solo subtipo.
- Son Exhaustivos, es decir, toda instancia de una superentidad es también una instancia de un subtipo.
- Es mutuamente exclusivo, lo que quiere decir que cada instancia de una superentidad es de un y solo un subtipo.

Arcos o Relaciones de Exclusividad:

Este tipo de relación expresa que no es posible que exista relación simultánea de una instancia de una entidad con las instancias de las entidades que participan en la relación de exclusividad o en el arco.

Es de anotar que se les llama arcos debido a su representación dentro del modelo. A continuación se presenta un ejemplo de uso de estas relaciones.



El ejemplo representa que una instancia de la entidad Factura le corresponde a una instancia de la entidad Persona o una instancia de Empresa, pero nunca a las dos al mismo tiempo.

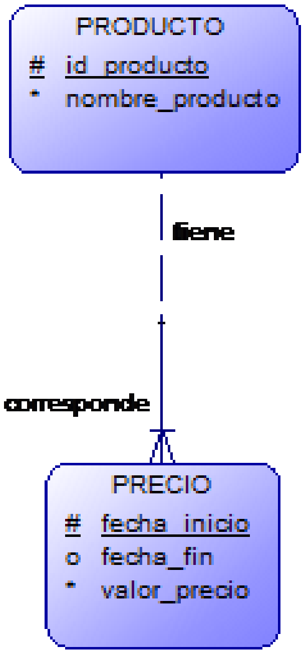
Se deben tener en cuenta algunas normas al momento de hacer uso de las relaciones exclusivas, estas son:

- Solo le pertenece a una entidad, es decir tal y como se presenta el ejemplo, si el arco se modelara en el lado contrario al del ejemplo, esto sería erróneo.
- Puede involucrar de dos a *n* relaciones
- No necesariamente debe excluir a todas las relaciones que llegan a la entidad, pero si siempre a mínimo dos.
- Las relaciones incluidas en el arco deben tener la misma opcionalidad.

1.2.1.5. Consideraciones Adicionales del Modelo ER

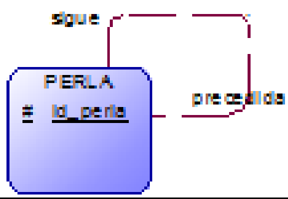
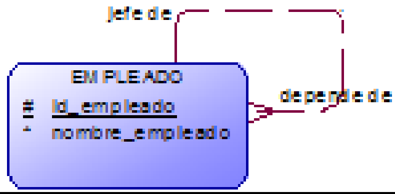
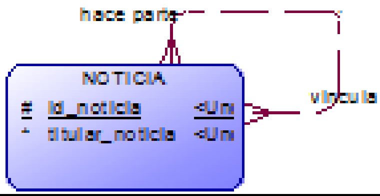
Modelando Cambios:

En muchos de los sistemas de la vida real necesitamos realizar el almacenamiento de los cambios que se pueden presentar en una entidad o en varias entidades, por ejemplo el cambio de precio de un Producto. Estos cambios se representan en el modelo haciendo uso de las entidades débiles.



Relaciones Recursivas:

Pese a que se mencionaron con anterioridad, es necesario especificar o presentar en qué casos se deben usar estas relaciones.

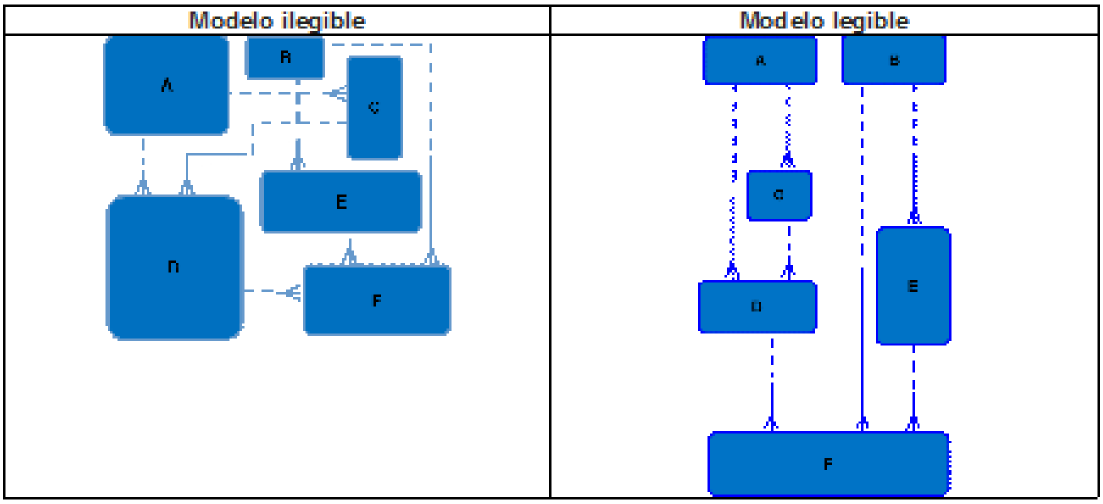
<table>
<tr><th>Relación</th><th>Descripción</th><th>Ejemplo</th></tr>
<tr><td>Uno a Uno</td><td>Se usa para expresar precedido por o seguido por.</td><td>
</td></tr>
<tr><td>Uno a Muchos</td><td>Expresa jefe de o dependiente de.</td><td>
</td></tr>
<tr><td>Muchos a Muchos</td><td>Expresa pares.</td><td>
</td></tr>
</table>

Es conveniente aclarar que aunque no existe una norma que lo establezca, se sugiere que las relaciones recursivas sean totalmente opcionales para evitar inconsistencias en el ciclo de vida, insistiendo que no es obligatorio que esto sea así.

Convenciones y legibilidad:

En término de entendimiento del modelo, se sugiere que las relaciones entre entidades se presenten en ángulos de noventa grados, adicionalmente que no se presenten cruces de las

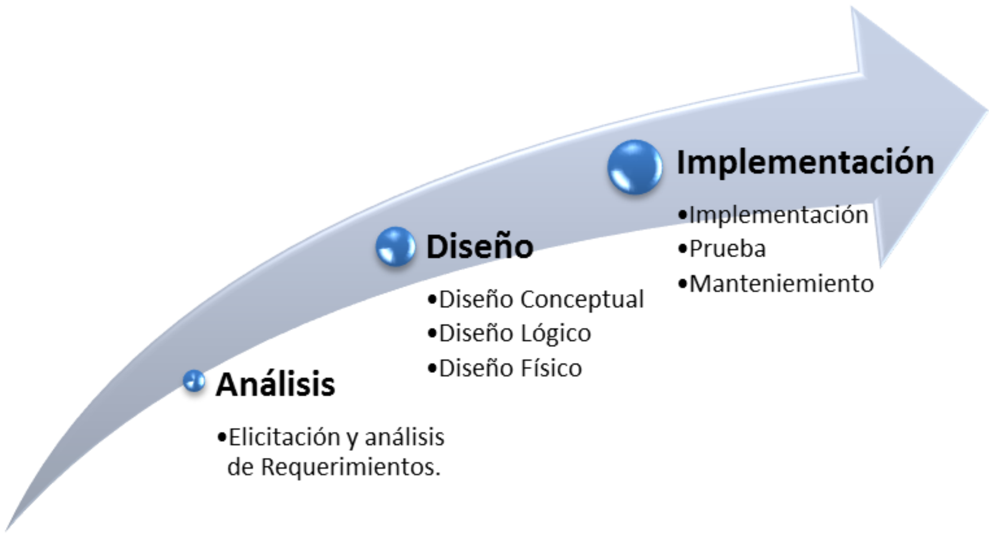
relaciones, pues dificultan la legibilidad del modelo.



1.3 Ciclo de Vida de las Bases de datos

El ciclo de vida de las bases de datos, como se mencionó anteriormente, inicia desde el análisis de las funcionalidades y datos que debe almacenar el sistema hasta llegar a la implementación, pero para llegar a este, es necesario pasar por una serie de etapas que si se llevan a cabo teniendo en cuenta parámetros de calidad, se llegara a tener una base de datos de una mayor funcional. En la figura 1.3 se presenta un esquema del ciclo de vida de una base de datos.

Figura 1.3: Ciclo de Vida de las Bases de datos



Como se mencionó en la introducción, este capítulo se centrara en la fase de Diseño, la cual es la más trascendental en el ciclo de vida de la base de datos, debido a que es en esta donde se representan, mediante el Modelo Entidad – Relación, todos los datos que se requieren para el desarrollo y adecuado funcionamiento del sistema.

1.3.1 Diseño Conceptual

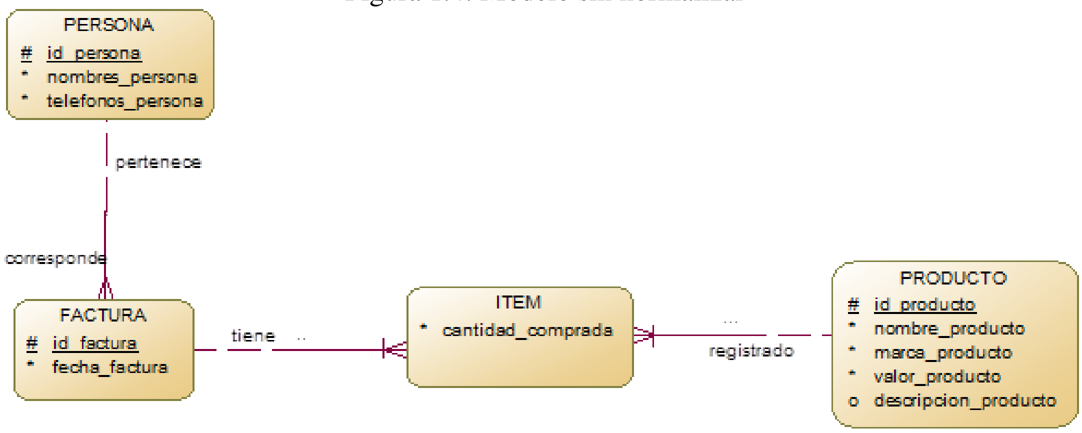
En el diseño conceptual se describe la información que se necesita para el negocio o sistema, facilitando la discusión con el usuario final para prevenir errores y mal entendidos. Es un aspecto importante en la documentación de un "sistema ideal" puesto que a través del tiempo se ha comprobado que es una forma una base sólida para el diseño físico de la base de datos.

El diseño o modelo conceptual pretende representar el "¿qué?" y no el "¿cómo?", por lo que es conveniente tener presente que en la fase conceptual aún no se debe pensar en el Sistema Gestor de Bases de Datos o con los recursos que cuenta la empresa.

A nivel conceptual se sugiere evaluar las posibles instancias de los atributos con el fin de determinar si estos pueden llegar a ser considerados como entidades con el fin de evitar redundancia. Lo anterior se conoce también como segunda forma normal dentro del proceso de normalización; cabe recordar que la primera forma normal corresponde a que los atributos de las entidades apliquen el principio de "Atomicidad", es decir, que no posean más de un valor dentro del mismo registro.

A continuación se presenta un ejemplo de un modelo en las formas normales mencionadas anteriormente:

Figura 1.4: Modelo sin normalizar



En el modelo de la Figura 1.4 se presentan algunos errores u omisiones, los cuales se solucionan al aplicar la primera forma normal. Dichas omisiones se evidencian en que algunos atributos incumplen el principio de la Atomicidad de las bases de datos (nombres_persona y telefonos_persona). Lo anterior se soluciona aplicando la primera forma normal (Ver Figura 1.5), aunque cabe aclarar que la definición de los atributos, es decir si son obligatorios u opcionales lo determina el CONTEXTO en el que se este planteando el modelo.

Una vez aplicada la primera normal, se debe revisar el modelo para evaluar que atributos tienen instancias que pueden repetirse continuamente. Para el caso del ejemplo presentado el atributo marca_producto en la entidad PRODUCTO puede presentar instancias reiterativas, por lo que para mayor funcionalidad este atributo se debe representar en el modelo como una entidad, lo anterior es conocido en términos más técnicos como que el atributo no depende funcionalmente del identificador de la entidad. Para este caso de la aparición de nuevas entidades, los atributos y relaciones que presenten vendrán dadas por la lógica del sistema.

1.3.2 Diseño Lógico

Para el caso del diseño o modelo Lógico, la variación con respecto al modelo conceptual corresponde principalmente a la inclusión de los atributos que almacenaran las relaciones; es aquí donde cabe resaltar la definición de **Llave Foránea**, la cual corresponde a un atributo que

Figura 1.5: Primera Forma Normal

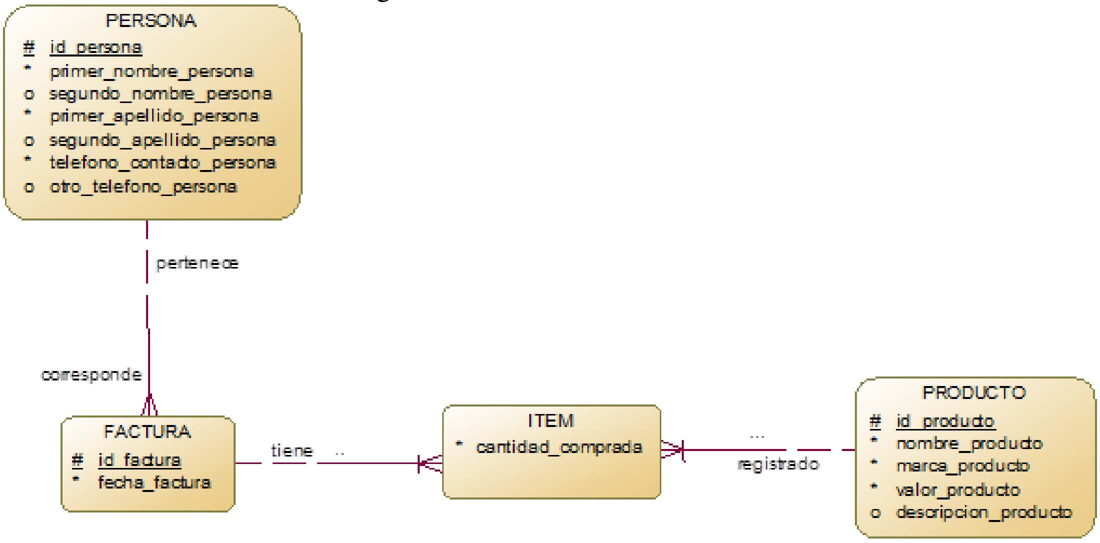
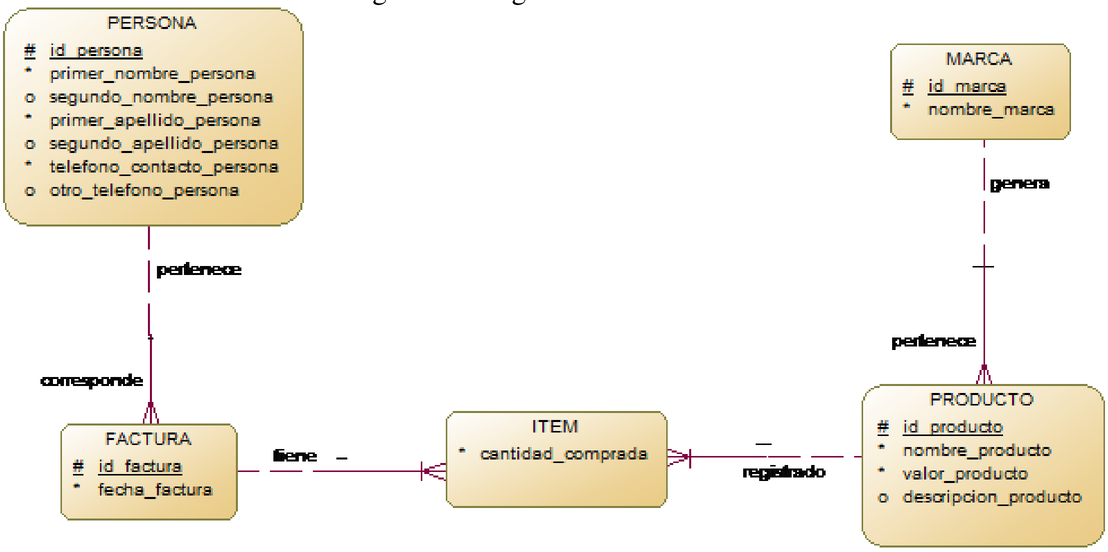


Figura 1.6: Segunda Forma Normal



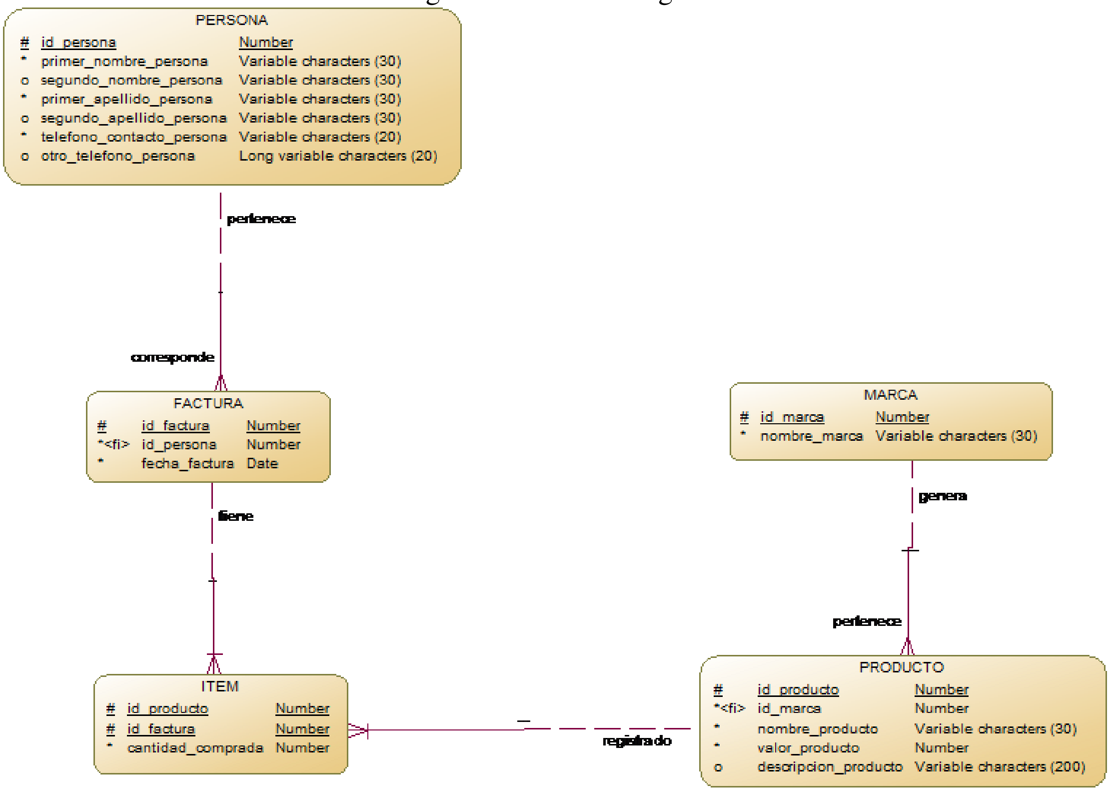
se genera en la entidad a la cual llegan las relaciones con grado muchos, o aquella en la que tenga más peso la relación, para el caso de las relaciones uno a uno. En caso de que en el modelo conceptual exista una relación muchos a muchos, esta se debe "romper" con una entidad débil así no tenga ninguna atributo que la represente.

Adicionalmente se sugiere definir los tipos de datos de los atributos de acuerdo al Sistema Gestor de Bases de Datos (SGBD) que se vaya a utilizar, teniendo en cuenta que no todos los SGBD incluyen los mismos tipos de datos.

Algunos autores sugieren pluralizar los nombres de las entidades desde el modelo lógico, aunque se puede realizar desde el modelo físico.

Para el caso de las entidades que incluyan subtipos, o subentidades, los atributos que contengan deberán definirse como opcionales, lo anterior con el fin de no afectar la generación y desarrollo de los modelos lógico y físico y por el principio de Exclusividad de los subtipos

Figura 1.7: Diseño Lógico



explicado con anterioridad.

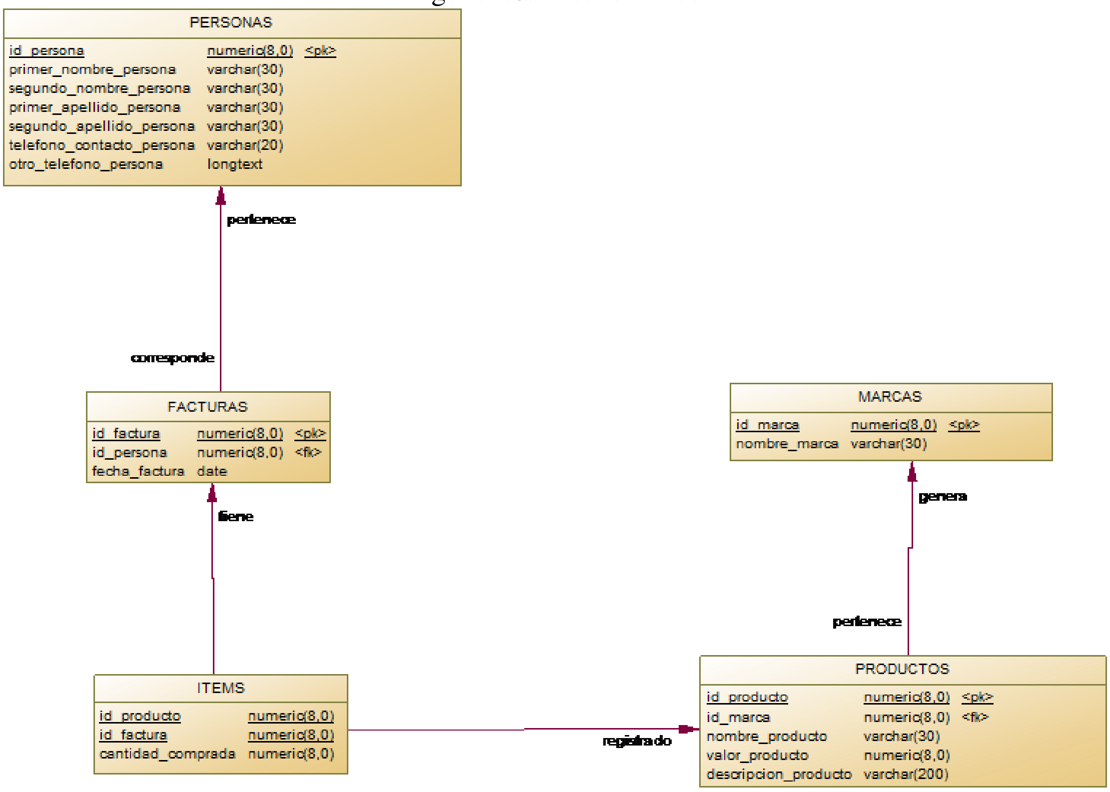
Se evidencia que para las entidades débiles, en el modelo lógico le adicionan dos atributos correspondientes a las relaciones existentes, pero adicionalmente dichos atributos son llaves primarias, esto no quiere decir que tenga dos llaves primarias, simplemente hace referencia a que se identificara por la combinación de los dos atributos, para el caso del ejemplo, a nivel de datos en la tabla ítem no podría existir una factura con el mismo producto más de una vez.

1.3.3 Diseño Físico

Finalmente, en el diseño físico, es donde se presenta el modelo correspondiente para la generación del código SQL para la creación de la base de datos en el SGBD seleccionado. Para el caso del Modelo Físico, en la mayoría de los casos, la notación utilizada no es Barker, pues desaparece la notación de "Pata de Gallina" en las relaciones, así como las líneas continuas y puntadas, y se usa notación Relacional, la cual es muy similar a UML (Unified Modeling Language, o Lenguaje Unificado de Modelado).

En el modelo se puede evidenciar que algunos tipos de datos varían con respecto a los manejados en los modelos conceptual y lógico, esto se debe a que en el modelo físico se deben manejar los tipos de datos propios del motor seleccionado (MySQL para el ejemplo). Y el otro cambio evidente es que los nombres de las entidades se presentan en plural, teniendo en cuenta que como se encuentren las entidades y atributos representados en el modelo físico es como se crearan en la base de datos.

Figura 1.8: Diseño Físico



1.4 Estándares de Diseño

Para el diseño existen múltiples estándares como la ISO/IEC 25012, la ISO/IEC 9126-3 y la ISO/IEC 25024, en el capítulo no se pretende dar una revisión minuciosa de cada una de las normas existentes, pues se requeriría todo un libro para esto, si no presentar las características más relevantes a tener en cuenta al momento de realizar el modelado de las bases de datos.

Figura 1.9: Estándares de Diseño

Estándar	Descripción
ISO/IEC 25012	Básicamente presenta quince (15) características de calidad, desde una vista inherente, cuyas características son la exactitud, completitud, consistencia, credibilidad y actualidad; y una vista dependiente del sistema, incluyendo la disponibilidad, portabilidad y recuperabilidad. El uso de la norma va principalmente orientado a determinar la calidad de los datos en las aplicaciones desarrolladas.
ISO/IEC 9126	Es un estándar enfocado en a las métricas internas de un producto de software, y en la parte 3 se definen las características de funcionabilidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad, las cuales son adaptadas por la ISO para el modelo de calidad de datos ISO/IEC 25012.
ISO/IEC 25024	A la fecha de redacción de este capítulo, es un estándar que aún se encuentra en desarrollo, pero por sus versiones técnicas se evidencia que está orientado netamente a proporcionar medidas para la calidad de los datos.

Adicional a los estándares mencionados, existen una serie de métricas que se sugieren sean revisadas con el fin de diseñar modelos de calidad y que sean realmente funcionales. Algunas de dichas métricas son:

- Métricas de Piattini
- Métricas de Moody
- Métricas de Kesh

1.5 Bibliografía

Atzeni, P., Ceri, S. & Paraboshi S. (1999). Database Systems. McGraw Hill.
Barker, R. (1990). CASE Method: Entity Relationship Modelling. Reading, MA: Addison-Wesley Pro
Garcia, H., Ullman, J. & Widom, J. (2012). DataBase Systems: The Complete Book. Prentice Hall.
González, M González, S. (2013). "Aplicación del estándar ISO/IEC 9126-3 en el modelo de datos conceptual entidad-relación" en Revista Facultad de Ingeniería. Vol. 22 No. 35

2 — Sistemas de Recuperación de Información (SRI)

Elaborado por: Cristina Bender - Claudia Deco
Universidad Nacional de Rosario
Universidad Católica Argentina, Campus Rosario

2.1 Introducción

2.1.1 Recuperación de Información

El objetivo principal de la Recuperación de Información es satisfacer la necesidad de información planteada por un usuario en una consulta en lenguaje natural especificada a través de un conjunto de palabras claves, también llamadas descriptores. En general, este proceso hacia la recuperación de documentos relevantes a la consulta presentada, no es un proceso simple debido a la complejidad semántica del vocabulario.

La Recuperación de Información o *InformationRetrieval* es la representación, almacenamiento, organización y acceso a ítems de información [Baeza et al., 1999]. Es un proceso mediante el cual se obtiene un conjunto de documentos que se adecuen a una demanda de información. La representación y organización de los ítems de información no son un problema simple de resolver, al igual que la caracterización de la necesidad de información del usuario tampoco lo es. Un ejemplo típico de un sistema de recuperación de información son los catálogos interactivos de las bibliotecas, donde una entrada del catálogo es ejemplo de un documento. Otro ejemplo es la web, donde cada página web representa un documento.

Un usuario puede desear recuperar un documento concreto o un conjunto de éstos. El usuario, debe traducir su necesidad de información en una consulta para luego ser procesada, en base a esta consulta, el objetivo primordial de un sistema de Recuperación de Información es recuperar información que sea útil o relevante para el usuario. Para la búsqueda, los usuarios suelen describir los documentos deseados mediante un conjunto de palabras claves. Por ejemplo, se puede utilizar la palabra clave "bases de datos relacionales" para buscar información sobre este tema.

Los documentos tienen un conjunto de palabras claves asociado. Los sistemas de recuperación de información recuperan aquellos documentos cuyos conjuntos de palabras claves contengan las proporcionadas por el usuario.

Entonces, un sistema de Recuperación de Información debe, de alguna manera, interpretar el contenido de los elementos de información o documentos de la colección y ordenarlos de acuerdo con el grado de relevancia para la consulta del usuario. La dificultad no está solo en conocer como extraer esta información sino también cómo utilizarla para decidir la relevancia de cada documento.

Según [Baeza et al., 1999], el modelo de Recuperación de Información se caracteriza formalmente como una cuádrupla (D , Q , F , R) donde D es una representación de la colección

de documentos; *Q* es una representación de la información que necesita el usuario (consulta); *F* es el entorno de trabajo para modelar la colección de documentos, las consultas y las relaciones que hay entre ellos; y *R*(*qi* , *dj*) es una función que devuelve un número real que permite asociar la consulta *qi* (*qi* pertenece a *Q*) y la representación de la colección de documentos *dj* (*dj* pertenece a *D*).

2.1.2 Recuperación de Información versus Recuperación de Datos

La meta principal de un sistema de Recuperación de Información es recuperar información que podría ser útil o importante para el usuario, y no sólo datos que satisfagan una consulta dada.

Un sistema de recuperación de *datos*, tal como una base de datos relacional, trata con datos que tienen una estructura y una semántica bien definidas. Un sistema de recuperación de datos permite recuperar todos los objetos que satisfacen las condiciones especificadas en una expresión regular o en una expresión del álgebra relacional. Por ejemplo, si se consulta por la palabra “cáncer” un sistema de recuperación de este tipo recuperará solamente aquellos objetos que contengan exactamente dicha palabra. Entonces, un sistema de recuperación de *datos* sólo recupera los datos que coinciden exactamente con el patrón ingresado por el usuario.

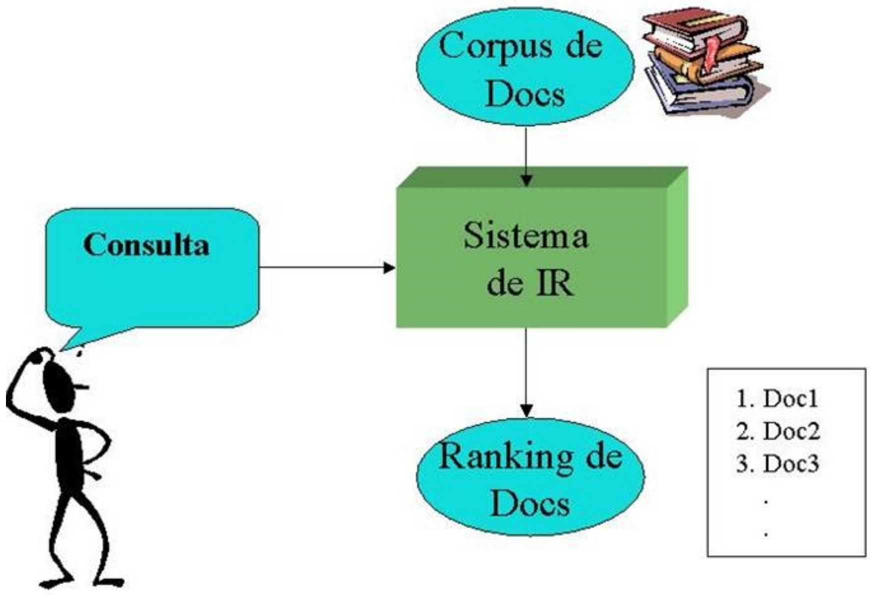
Un sistema de recuperación de *información* recupera datos relevantes que hagan la mejor coincidencia parcial con el patrón dado. Esto se debe a que la recuperación de información generalmente trata con texto en lenguaje natural, el cual no está siempre bien estructurado y podría ser semánticamente ambiguo. Por ejemplo, si se realiza una consulta por el término “cáncer” en un sistema de recuperación de información, además de obtener como resultado los documentos que contengan este término, se debería obtener también los documentos en que aparezca ”neoplasma”, ”carcinoma”, “cancerígeno”, etc..

[Blair, 1990: 2-4] clasifica las diferencias entre recuperación de datos (*data retrieval*) y recuperación de información (*information retrieval*) de la forma siguiente:

	Recuperación de datos	Recuperación de información
Según la forma de responder a la pregunta	se utilizan preguntas altamente formalizadas, cuya respuesta es directamente la información deseada	las preguntas resultan difíciles de trasladar a un lenguaje normalizado, y la respuesta es un conjunto de documentos que pueden contener, sólo probablemente, lo deseado, con un evidente factor de indeterminación
Según la relación entre el requerimiento al sistema y la satisfacción del usuario	la relación es determinística entre la pregunta y la satisfacción	la relación es probabilística, a causa del nivel de incertidumbre presente en la respuesta
Según el criterio de éxito	el criterio a emplear es la corrección y la exactitud	el único criterio de valor es la satisfacción del usuario, basada en un criterio personal de utilidad
Según la rapidez de respuesta	depende del soporte físico y de la perfección del algoritmo de búsqueda y de los índices	depende de las decisiones y acciones del usuario durante el proceso.

2.1.3 Componentes de un Sistema de Recuperación de Información

En la figura siguiente se grafican los componentes de un sistema de recuperación de información.



Se posee una colección o cuerpo de documentos. El sistema de recuperación de información (Sistema de IR en la figura) trabaja con estos documentos realizando operaciones sobre los textos, tales como remoción de palabras no significativas y *stemming*; para construir un índice invertido (o archivo invertido) de palabras con punteros a los documentos. Este tema se amplía en el Apartado 2.6.

Una Interface de usuariomaneja la interacción con el usuario permitiéndole el ingreso de la consulta (*query*), y la visualización de los resultados. El Sistema de Recuperación de Información realiza operaciones de transformación de la *query* para mejorar la recuperación, por ejemplo, la expansión de la consulta utilizando recursos lingüísticos o usando *feedback* de relevancia. Así, el sistema recupera los documentos que contienen los términos que están en el índice invertido. Este tema se amplía en el Apartado 2.5.

El resultado se muestra al usuario con un orden (*ranking*) de todos los documentos recuperados de acuerdo a una métrica de relevancia. Este tema se amplía en el Apartado 2.4.

Por lo recién descripto, existe un conjunto de tareas involucradas en los sistemas de recuperación de información que permiten cumplir con el objetivo de estos sistemas. Estas tareas son: la categorización (automática) de documentos para poder clasificarlos; el filtrado de información que permite descartar la información no relevante; el clustering automático de documentos para poder generar grupos de documentos afines; la extracción de información; y la integración de información.

De aquí, se advierte la existencia de varias áreas relacionadas con la recuperación de información. A continuación se describe brevemente la incumbencia de cada área relacionada en los aportes que brinda a la Recuperación de Información.

2.1.4 Areas relacionadas con la Recuperación de Información

La primer área de interés es el gestionamiento de Bases de Datos (*Database Management*) queseenfoca en los datos estructurados almacenados en tablas relacionales más que en texto sin formato; y se focaliza en el procesamiento eficiente de consultas bien-definidas en un lenguaje formal (SQL). También tiene una semántica clara para los datos y las consultas. Recientemente se ha volcado a los datos semi-estructurados (XML) y esto lo ha llevado más cerca de la Recuperación de Información.

El área de las Ciencias de la Información (*Library and Information Science*) se focaliza en los aspectos del usuario humano de la Recuperación de Información, entre los que se destacan la

interacción humano-computadora, la interface de usuario, la visualización. Esta área se enfoca en la preparación de una estrategia de búsqueda. Los temas concernientes a esta área son el análisis de citación y la bibliometría. Los trabajos recientes sobre bibliotecas digitales acercan a esta área a las áreas de Ciencias de la Computación y de Recuperación de Información.

La Inteligencia Artificial (*Artificial Intelligence*) se focaliza en la representación del conocimiento, razonamiento y acción inteligente. Utiliza formalismos para representar el conocimiento y las consultas; por ejemplo la lógica de predicados de primer orden. El trabajo reciente en ontologías de la web y agentes de información inteligentes la lleva más cerca de la Recuperación de Información.

El Procesamiento de Lenguaje Natural (*Natural Language Processing*) se focaliza en el análisis sintáctico y semántico de texto en lenguaje natural. La habilidad para analizar sintaxis (esto es la estructura de la frase) y semántica (esto es significado de la frase) podría permitir la recuperación basada en significado más que en *keywords*. Esta área desarrolla métodos para determinar el sentido de una palabra ambigua basada en su contexto (WSD - word sense disambiguation), y desarrolla métodos para identificar piezas específicas de información en un documento (esto es: colabora con la tarea de extracción de información).

El *Machine Learning* se focaliza en el desarrollo de sistemas computacionales que mejoren su performance con la experiencia. Permite realizar la clasificación automática de documentos basada en conceptos aprendidos a partir de ejemplos etiquetados de entrenamiento. Colabora en la categorización de texto (*Text Categorization*) utilizada, por ejemplo, por Yahoo para realizar la clasificación automática de jerarquías; y en el agrupamientode texto (*Text Clustering*), para el agrupamiento de resultadosde una consulta de la Recuperación de Información. Otra actividad que facilita es el minado de textos (*Text Mining*).

2.1.5 Recuperación de Información en la Web

Los motores de búsqueda recolectan páginas de la web, las indexan, buscan en los índices las palabras claves ingresadas en la consulta, utilizan algoritmos de ranking para ordenar los resultados y muestran al usuario los documentos resultantes.
Una página web corresponde a un documento en la recuperación de información tradicional. La recuperación de información en la web considera como una colección de documentos la parte de la web que está públicamente indexada, excluyendo las páginas que no puedan ser indexadas por ser muy dinámicas o por ser privadas.
Los motores de búsqueda están potenciados por técnicas de recuperación de información. Algunos ejemplos de estos motores de búsqueda son: Google (`www.google.com`), Yahoo! (`www.yahoo.com`), Bing (`www.bing.com`), entre otros.
Uno de los problemas que surgen con los motores de búsqueda de la web es que los usuarios no tienen el tiempo y el conocimiento para seleccionar el o los motores más adecuados para su necesidad de información. Una solución posible a esto son los motores de meta búsqueda, tal como MetaCrawler (`www.metacrawler.com`), que son servidores web que envían la consulta a varios motores de búsqueda; recopilan estos resultados y los unifican, uniéndolos y presentándoselos a los usuarios.

2.2 Modelos de Recuperación de información

Un modelo de recuperación de información es una idealización o abstracción del proceso real de recuperación. Dentro de un modelo pueden identificarse diferentes partes: Un modelo para la representación de documentos (datos), un modelo para representar las consultas que los usuarios pueden formular (que sería un análogo al lenguaje de consultas de un modelo de base de datos), un entorno de modelado de las relaciones entre documentos y consultas y una función

de ranking $R(d,q)$, que asocia un número real con una consulta q y un documento d. El ranking es una estimación cuantitativa de la probabilidad de que el documento d resulte relevante para la consulta q.

En un sistema de recuperación de información habitualmente no se trabaja con los documentos propiamente dichos, sino con una representación más manejable de los mismos. Un documento puede representarse por una colección de valores de ciertos campos como el título del documento, sus autores, fecha de publicación, etc., como es en el caso del catálogo de una biblioteca. Otra posibilidad, utilizada en los modelos básicos que se presentan a continuación, es describir los documentos a través de un conjunto de términos representativos, llamados *keywords*, que describen de alguna forma el contenido de un documento y no necesariamente deben aparecer como una palabra o una frase del documento mismo. Sin embargo, se pueden considerar todos los términos que aparecen en el documento como *keywords* en una aproximación llamada '*full text*'. En este caso, se obtiene una representación del documento llamada bolsa de palabras, en la que el documento es representado como una lista desordenada de todas las palabras que aparecen en él.

No todas las palabras tienen la misma importancia dentro de un documento y una forma de aprovechar esto es asignar pesos a las palabras para medir su importancia para un dado documento. La función peso $W(Ti,dj)$, asigna un número real mayor o igual a cero a cada par (Ti,dj) para medir la importancia del término Ti en el documento dj.

En este capítulo se describen tres modelos básicos de recuperación de información: el modelo Booleano, el modelo Espacio Vectorial y el modelo Probabilístico. Modelos más avanzados, algunos de los cuales utilizan técnicas de inteligencia artificial para representar ciertas características de los documentos, pueden consultarse en [Baeza et al., 1999], [Salton, 1983], [van Rijsbergen, 1979], entre otros.

2.2.1 Modelo Booleano

El modelo de recuperación de información Booleano está basado en la teoría clásica de conjuntos y el álgebra de Boole. En este modelo, los documentos se representan por el conjunto de términos contenidos en ellos. Así, la representación de la colección de documentos se realiza sobre una matriz binaria documento-término, donde los términos han sido extraídos manualmente o automáticamente de los documentos y representan el contenido de los mismos.

Las consultas se expresan como expresiones booleanas entre términos, usando los operadores AND, OR y NOT. Una desventaja del modelo es que limita su aplicación a aquellos usuarios que tengan conocimientos apropiados para manejar expresiones lógicas. La semántica de los operadores booleanos en una consulta puede describirse informalmente de la siguiente forma:
T1 AND T2 = conjunto de documentos cuyas representaciones contienen al término T1 y al término T2.
T1 OR T2 = conjunto de documentos cuyas representaciones contienen al término T1 o al término T2.
NOT T1 = conjunto de documentos cuyas representaciones no contienen al término T1.

Para el modelo booleano, los pesos de los términos son binarios, esto es: $W(Ti,dj)$ pertenece a $\{0,1\}$, siendo el peso del término Ti igual a uno si este pertenece al documento dj y cero en caso contrario.

La similitud $R(dj, q)$ de un documento dj y una consulta q valdrá 1 si los términos contenidos en la representación del documento dj hacen verdadera a la expresión de la consulta q, y valdrá 0 en caso contrario. Este ranking binario es la principal desventaja del modelo, puesto que dada una consulta, un documento o es relevante a la misma o no lo es, y el conjunto de documentos que se obtiene como resultado de una consulta no puede ser ordenado por un ranking de relevancia, ya que todos ellos son igualmente relevantes. Sin embargo, como ventajas de este modelo se tiene

que posee un formalismo muy simple y una semántica clara y concisa para la formulación de las consultas, y fue adoptado por muchos de los primeros sistemas de recuperación de información.

2.2.2 Modelo Espacio Vectorial

En el modelo Espacio Vectorial, tanto los documentos como las consultas se representan por vectores de términos. Cada documento es representado por un vector T-dimensional, donde T es el número total de términos en la colección. De la misma manera, una consulta se representará por un vector T-dimensional. Cada elemento del vector representará el peso que se le asigne al término correspondiente a ese elemento en el documento (o consulta). El modelo vectorial utiliza pesos no binarios para los términos de los documentos para así poder computar el grado de similitud entre documentos y consultas de forma gradual. Esto permite que el conjunto de documentos obtenidos como resultado de una consulta pueda ser ordenado por un ranking de relevancia.

Un esquema de pesado habitual dentro de este modelo está basado en consideraciones estadísticas para intentar representar la importancia relativa de un término dentro de un documento. Se tiene en cuenta la cantidad de veces que aparece una dada palabra en un documento, suponiendo que las palabras que se repiten más veces son más representativas del contenido del mismo. La frecuencia de un término *Ti* en el documento *dj*, notada *F*(*Ti*,*dj*), es el número de veces que aparece el término *Ti*en el documento *dj*.

Un problema de considerar como peso de una palabra directamente a su frecuencia de aparición es que se le confiere igual importancia a todos los términos que aparecen en la colección. Si se supone que los términos que aparecen en pocos documentos son buenos discriminadores y que los términos más comunes son menos útiles a la hora de decidir la relevancia de un documento, es natural asignar a los primeros un peso más alto que a los segundos. La frecuencia de documento inversa (IDF: *Inverse Document Frecuency*) de una palabra está relacionada con la cantidad de documentos de la colección en que aparece dicha palabra. Como el término se considera más importante cuando aparece en menos documentos, la frecuencia de documento inversa se define como: $IDF(Ti) = log(N/ni)$ donde N es la cantidad total de documentos y ni es el número de documentos en los que aparece el término *Ti*. El logaritmo se incluye sólo para evitar el crecimiento numérico de la función. Una palabra que aparezca en todos los documentos de la colección tendrá entonces una *IDF* igual a cero, indicando que carece totalmente de valor para discriminar documentos.

El peso de un término *Ti* dentro de un documento *dj* se define entonces de la siguiente manera: $W(Ti,dj) = TF(Ti,dj) * IDF(Ti)$. Este esquema de pesado se denomina *TF-IDF* y es uno de los más utilizados dentro de los sistemas de recuperación de información construidos utilizando el modelo Espacio Vectorial. Existen otros sistemas de asignación de pesos, y el lector interesado puede consultarlos, por ejemplo, en [Baeza et al., 1999].

Las consultas en el modelo Espacio Vectorial pueden ser expresadas en un lenguaje cercano al natural, como una simple enumeración de palabras. Al no necesitar la utilización de operadores explícitos como en el modelo Booleano, el sistema resulta más accesible y más natural de usar para un usuario sin conocimientos específicos sobre el tema.

La similitud entre dos documentos o entre un documento y una consulta se define en una primera aproximación como el producto interno de los vectores que los representan.

$$R(\overline{q},\overline{d}) = \overline{q} \bullet \overline{d} = \sum_{I=1}^{T} (q_i \times d_i)$$

Con pesos binarios, esto equivale a contar el número de términos coincidentes entre ambos documentos, o entre un documento y una consulta. Utilizando el esquema de asignación de peso

TF-IDF, la importancia de cada término influirá en la medida del ranking. Por ejemplo, dada una consulta, un documento con una única coincidencia, pero en un término con un peso alto, podría tener un ranking más elevado que otro con varias coincidencias en términos de pesos pequeños. Una medida más usual de similitud es la medida del coseno:

$$cos\,(\Theta) = S(\bar{q},\bar{d}) = \frac{\bar{q} \bullet \bar{d}}{\|\bar{q}\| \times \|\bar{d}\|} = \frac{\sum_{i=1}^{T} (q_i \times d_i)}{\|\bar{q}\| \times \|\bar{d}\|}$$

que se puede interpretar como el coseno del ángulo formado por los vectores representativos de cada documento *di* y la consulta q en el espacio T-dimensional. La similitud será entonces un número real entre 0 (ninguna coincidencia) y uno (mayor coincidencia). El modelo espacio vectorial es el modelo más difundido actualmente.

Veamos un ejemplo. Supongamos que tenemos un documento que comienza con el texto siguiente: *"La República Argentina ha sido nominada para la realización del X Congreso Americano de Epidemiología en Zonas de Desastre. El evento se realizará ..."*. Un usuario realiza la siguiente consulta: *"argentina congreso epidemiología"*. En la tabla siguiente se observa parte de la matriz término-documento con pesos normalizados entre 0 y 1. En la última fila de la tabla se presenta la representación de la consulta.

	argentina	**...**	**congreso**	**epidemiología**	**...**
d_1	0.5		0.3	0.2	
...					
d_n					
consulta	0.4		0.3	0.3	

En la figura siguiente se representan gráficamente el documento (**D**) y la consulta (**C**), y la similitud entre la consulta y el documento se mide con el valor del coseno entre ambos vectores (el arco en rojo en la figura).



2.2.3 Modelo Probabilístico

El modelo probabilístico de recuperación de información está basado en la teoría de probabilidades. Dada una consulta q y un documento d, el modelo intenta determinar la probabilidad de que el documento d resulte relevante a la consulta. El modelo asume que esta probabilidad de relevancia depende sólo de las representaciones del documento y la consulta.

En el modelo probabilístico los documentos se representan por el conjunto de términos que contienen, donde los pesos de los términos son binarios $W(Ti,dj)$ pertenece a $\{0,1\}$. Las consultas se expresan como una enumeración de términos.

Una suposición básica que realiza este modelo es: dada una consulta, existe exactamente un conjunto de documentos, y no otro, que satisface dicha consulta. Este conjunto es llamado el conjunto ideal. El conjunto ideal no se conoce de antemano, y es necesario realizar ciertas suposiciones sobre las propiedades de dicho conjunto, que se intentan refinar consulta tras consulta. Tras cada consulta, el usuario identificará los documentos recuperados que resultaron relevantes, con lo que se refinará la descripción del conjunto ideal.

La similitud entre un documento dj y una consulta q se define como:

$$sim(d_j, q) = \frac{P(R|d_j)}{P(R'|d_j)} = \frac{\dfrac{P(d_j|R) \times P(R)}{P(d_j)}}{\dfrac{P(d_j|R') \times P(R')}{P(d_j)}} = \frac{P(d_j|R)}{P(d_j|R')}$$

donde:
P(R|dj) es la probabilidad de que el documento dj sea relevante a la consulta q.
P(R'|dj) es la probabilidad de que el documento dj no sea relevante a la consulta q.
P(dj|R) es la probabilidad de seleccionar al documento dj de entre los relevantes
P(R) es la probabilidad de que seleccionando algún documento aleatoriamente de la colección, sea relevante.
P(dj) es la probabilidad de obtener el documento dj aleatoriamente seleccionando uno de entre toda la colección.
P(R'|dj), P(dj |R'), P(R') son los análogos, aplicados a la no relevancia

Entonces, un documento dj será considerado como relevante si:
P(R|dj) > P(R'|dj) o P(dj|R) > P(dj|R')

De esta manera el modelo permite ordenar los documentos de la colección en orden descendente de probabilidad de relevancia en relación a la consulta, superando así la gran deficiencia del modelo booleano, a pesar de seguir siendo un modelo binario. Este modelo tiene desventajas ya que necesita una hipótesis inicial para establecer los documentos relevantes y el peso de sus términos, y además, supone independencia entre términos.

2.2.4 Otros modelos

Existen otros modelos para representar la recuperación de información, tales como el modelo de Conjuntos Difusos (*Fuzzy Sets*), apoyado en conceptos de la teoría difusa; el modelo Booleano Extendido; el modelo Espacio Vectorial Generalizado; el modelo de Indexado Semántico Latente; el modelo de Red Neuronal; modelos probabilísticos alternativos, que consideran el uso de redes bayesianas y redes de inferencia; modelos de recuperación de texto estructurado; y modelos para la visualización (*browsing*). El lector interesado puede profundizar estos temas en [Baeza et al., 1999].

2.3 Lenguajes de consulta

Una *consulta* en un sistema de recuperación de información es una solicitud de documentos pertenecientes a algún tema. Dada una colección de documentos y una consulta del usuario, el

objetivo de una *estrategia de búsqueda* es obtener todos y sólo los documentos relevantes a la consulta.

Para iniciar un proceso de recuperación de información el usuario debe hacer una solicitud al sistema a través de una consulta. Cada modelo de recuperación de información responderá a distintos tipos de consultas. En los tres modelos básicos descriptos en el apartado anterior las consultas están basadas en palabras clave o *keywords*. En algunos modelos más avanzados las consultas se formulan en lenguaje natural y se resuelven aplicando técnicas de Procesamiento de Lenguaje Natural.

2.3.1 Consultas basadas en *keywords*

Las consultas basadas en *keywords* se componen de una combinación más o menos compleja de palabras que se emplearán para determinar qué documentos son relevantes. Son intuitivas y fáciles de expresar, ya que no requieren que el usuario conozca un formalismo para construir su solicitud. Permiten un ranking rápido determinado por la cantidad de keywords de la consulta original que pertenecen a la representación del documento retornado junto a lo representativa del contenido que sea cada una de ellas. Las consultas basadas en *keywords* se pueden clasificar en consultas de una palabra, consultas contextuales, consultas booleanas y consultas en lenguaje natural. Las consultas de una palabra son aceptadas por todos los modelos, el resultado de la consulta es el conjunto de documentos cuyas representaciones contienen la palabra buscada. El ranking se elabora sobre medidas estadísticas como la frecuencia de términos y la frecuencia de documentos inversa.

2.3.2 Consultas contextuales

Las consultas contextuales constan de varias palabras y la búsqueda se realiza teniendo en cuenta el contexto del documento en que aparecen las palabras buscadas. Algunos ejemplos de consultas contextuales son la búsqueda de frases y las búsquedas por proximidad. La idea de las consultas por proximidad es lograr recuperar aquellos documentos en que las palabras de la consulta aparezcan citadas en un mismo contexto. Este contexto se determina midiendo la proximidad entre palabras. Una suposición que se realiza es que las palabras que estén muy próximas entre sí en una dada parte del documento se referirán a un mismo tema. Dada una unidad de medida de distancia entre términos, usualmente la cantidad de palabras o párrafos que hay entre ellos, y algunas características de esa distancia, como si debe ser unidireccional o bidireccional según la distancia se mida solo hacia adelante o hacia adelante y atrás en el texto; se retornarán los documentos donde las palabra de la consulta aparezcan suficientemente próximas entre sí. En las búsquedas de frases las palabras deben aparecer dentro del documento en posiciones consecutivas y en el mismo orden que en la frase de la consulta. Se puede ver a este tipo de consultas como un caso particular de consultas de proximidad en que se exige que la proximidad entre las palabras debe ser la máxima posible.

El ranking de consultas contextuales se puede determinar según la proximidad entre los términos en los documentos recuperados, asignando mayor ranking a los documentos en que las palabras aparezcan más próximas.

Las búsquedas contextuales proporcionan un mecanismo para identificar aquellos documentos en los cuales los diferentes términos de la consulta se refieren a un mismo tema, contando únicamente con información sobre la posición de las palabras dentro de los documentos, evitando asi retornar resultados vagos de documentos en los que solo se mencionan los términos en diferentes contextos. Por ejemplo, si en una consulta de proximidad aparecen las palabras "cáncer" y "cerebro", un documento que hable de Medicina en general y que cite a "cerebro" en una enumeración de los órganos del cuerpo humano en una sección, y "cáncer" en una lista de enfermedades en otra sección, tendrá menor proximidad y por lo tanto peor ranking que uno

donde aparezcan ambos términos en un mismo párrafo.

2.3.3 Consultas booleanas

Las consultas booleanas se expresan dando los términos separados por operadores booleanos explícitos (AND, OR, NOT). Se resuelven aplicando las operaciones de conjuntos correspondientes (intersección, unión, complemento o diferencia, respectivamente) sobre los conjuntos de documentos que contienen a los términos de la consulta en sus representaciones.

Los sistemas booleanos puros manejan sólo la información de si una palabra es *keyword* o no de un documento, y como se dijo antes, esta información no es suficiente para determinar un ranking de los resultados.

La sintaxis de una consulta booleana está compuesta por átomos, que junto con los operadores booleanos recuperan documentos. Se define un árbol de consulta donde las hojas corresponden a las preguntas básicas y los nodos internos a los operadores (OR, AND, NOT).

2.3.4 Consultas en lenguaje natural

Las consultas en lenguaje natural son simples enumeraciones de términos sin emplear operadores, y por lo tanto presentan una interfaz amigable al usuario común. A diferencia de las consultas booleanas permite recuperar documentos que satisfagan sólo parcialmente la consulta. El ranking de los resultados se puede elaborar teniendo en cuenta el número de partes de la consulta que se satisfacen, mientras más términos de la consulta estén presentes mejor ranking tendrá el documento. Es directamente aplicable al modelo de Recuperación de Información de espacio vectorial.

El lenguaje natural, esto es la herramienta que utilizan las personas para expresarse, posee dos propiedades que merman la efectividad de los sistemas de Recuperación de Información. Estas propiedades son la variación lingüística y la ambigüedad lingüística. La variación lingüística es la posibilidad de utilizar diferentes palabras o expresiones para comunicar una misma idea. La ambigüedad lingüística se refiere a cuando una palabra o frase permite más de una interpretación. Ambas propiedades inciden en el proceso de recuperación de información aunque de forma distinta. La variación lingüística provoca el silencio documental, es decir la omisión de documentos relevantes para cubrir la necesidad de información, ya que no se han utilizado los mismos términos que aparecen en el documento. En cambio, la ambigüedad implica el ruido documental, es decir la inclusión de documentos que no son significativos, ya que se recuperan también documentos que utilizan el término pero con significado diferente al requerido. Estas dos características dificultan considerablemente el tratamiento automatizado del lenguaje.

Por ejemplo, a nivel morfológico una misma palabra puede adoptar diferentes roles morfosintácticos en función del contexto en el que aparece, ocasionando problemas de ambigüedad. Consideremos la siguiente frase: "*Deja la comida que sobre sobre la mesa de la cocina, dijo llevando el sobre en la mano.*", donde la palabra *sobre* es ambigua morfológicamente ya que puede ser un sustantivo masculino singular, una preposición, y también la primera o tercera persona del verbo sobrar, según el orden en que aparece en la frase dada.

2.3.5 Otros tipos de consultas

Existen otros tipos de consultas, que no están basadas en palabras, como por ejemplo las consultas basadas en reconocimiento de patrones (*pattern matching*) que permiten búsquedas parciales de términos. Se determinan qué documentos recuperar según contengan porciones de texto que satisfagan alguna propiedad (patrón). Los patrones pueden ser prefijos, esto es que el texto contenga palabras con el prefijo dado, como (bio*); sufijos, es decir que aparezcan los sufijos dados dentro del texto, por ejemplo (*logía); rangos, para buscar documentos que contengan valores, como números, fechas o caracteres, en determinado rango ($1901<x<2004$);

o expresiones regulares que permiten formular la consulta como una expresión regular y se recuperan aquellos documentos que contengan subcadenas que las satisfagan. Para las expresiones regulares las operaciones más comunes son unión (a|b), la concatenación (a b) y la repetición (a*). El lector interesado en este tema, puede ampliarlo en [Baeza et al., 1999], [van Rijsbergen, 1979], entre otros.

2.4 Evaluación de la Recuperación de Información

2.4.1 Relevancia

Un problema importante de la Recuperación de Información es el grado de relevancia de los documentos. Un sistema de recuperación de información para ser efectivo, debe en alguna forma interpretar los contenidos de los documentos en una colección y ordenar los resultados según un ranking de acuerdo al grado de relevancia que tenga respecto a la consulta del usuario. Por lo tanto, un objetivo de la Recuperación de Información es recuperar todos los documentos que sean relevantes a una consulta del usuario y recuperar la mínima cantidad de documentos no relevantes.

Existen dos tendencias a la hora de definir relevancia. La relevancia objetiva,que hace hincapié en los sistemas, normalmente define cómo el tema de la información recuperada coincide con el tema de la pregunta.

La relevancia subjetiva es la que tiene en cuenta al usuario [Swanson, 1988]. Es un juicio subjetivo y puede incluir: que corresponda al tema buscado, que sea actual (información reciente), que provenga de una fuente confiable, y que satisfaga los objetivos del usuario y su necesidad de información. Para [Schamber, 1990] la relevancia se refiere a la utilidad, o potencial uso de los materiales recuperados, con relación a la satisfacción de los objetivos, el interés, el trabajo o los problemas intrínsecos del usuario. En la relevancia subjetiva, se estudia desde el punto de vista de la información nueva que consigue un usuario de un documento. Según este concepto, la información conocida no es relevante.

Muy ligado al concepto de relevancia está el de pertinencia; con frecuencia se entremezclan y confunden. En general, relevancia es la medida de cómo una pregunta se ajusta a un documento (relevancia objetiva) y pertinencia es la medida de cómo un documento se ajusta a una necesidad informativa (relevancia subjetiva).

Para dar un ejemplo consideremos a un usuario que desea obtener una descripción sobre el automóvil Volkswagen Golf. Si plantea la consulta de forma demasiado simple ingresando sólo la palabra “Golf”, además de los documentos de interés buscados, va a recuperar también aquellos que sólo mencionan al automóvil, como noticias o avisos clasificados; o que contienen el término “Golf” con otro significado, como puede ser una noticia sobre el deporte, o una receta de cocina que lleve salsa golf. Si el usuario refina su pedido agregando más términos que describan su dominio de interés, extendiendo la consulta a “automóvil Golf”, obtendrá más información sobre el VW Golf, pero no recuperará aquellos documentos útiles en los que no aparezca la palabra “automóvil” y que empleen sinónimos como “auto”, “vehículo”, “carro”, “automobile” o “car”.

Cualquiera sea el caso, los usuarios no pueden revisar exhaustivamente todos los resultados recuperados hasta hallar aquellos que se ajusten a sus necesidades y es necesario que el sistema provea algún tipo de asistencia automática. Una solución es presentar los resultados de forma ordenada mediante un ranking de relevancia. De esta manera se pueden presentar todos los documentos que satisfacen la consulta, aumentando el recall, pero dando mejor posición a los documentos que satisfacen la consulta con mayor exactitud. De esta manera se los separa en cierta manera del resto, aumentando la precisión. Este ranking facilita la búsqueda entre los resultados al usuario, permitiéndole reducirla a revisar unos cuantos documentos ubicados en las primeras posiciones, donde posiblemente hallará los más relevantes.

Existen diferentes formas de elaborar un ranking de relevancia, por ejemplo, midiendo la similitud entre los documentos y las consultas. Los diferentes aspectos que pueden tenerse en cuenta en la construcción de un ranking pueden clasificarse en internos y externos, según utilicen propiedades propias de los documentos o propiedades externas a ellos.

Los aspectos internos influirán en la importancia que se le otorgará a las palabras dentro de un documento. Además de las consideraciones estadísticas ya mencionadas, como la frecuencia de palabras, otros aspectos que pueden considerarse son la ubicación de los términos dentro del documento, concediéndole mayor importancia a aquellos que aparezcan en títulos o subtítulos; el tipo de fuente, dando más relevancia a las palabras resaltadas, en negritas, o en cursiva; etc. Los aspectos externos determinarán una medida de la importancia del documento dentro de la colección. Entre algunas propiedades externas que pueden ser empleadas para realizar el Ranking de Relevancia se encuentran las citas hechas por otros documentos, la ubicación del documento dentro de la colección, y la popularidad de los mismos.

Un ranking de citas es una medida de la importancia de un documento dentro de una colección basada en la cantidad de documentos que lo citan. En el caso de páginas web, se puede utilizar en forma similar los enlaces entre páginas. Un ejemplo es el Page-Rank [Page et al., 1998] utilizado en el buscador web Google, que asigna un ranking a los documentos basándose en la cantidad de páginas que contienen enlaces hacia él. Otras características de las citas que aportan al ranking del documento son la importancia de los documentos que hacen las citaciones, la relevancia del texto circundante a las mismas y la relevancia de esos documentos.

En el caso de la web, otra propiedad externa que puede considerarse es la ubicación del documento en el árbol de directorios, si se halla en un directorio importante o cerca del directorio raíz figurará con mejor puntaje. La popularidad de consultas similares es otra propiedad, como es el caso de buscadores web que rastrean los clicks de los usuarios sobre las direcciones retornadas para cada consulta, para hacer un ranking de relevancia.

Estos diversos factores se combinan para otorgar un ranking de relevancia a los resultados de una consulta. En general siempre se tiene en cuenta la similitud entre la consulta y los documentos, dependiendo del sistema en particular la importancia que se le dará a los diferentes factores en el cálculo del ranking final.

2.4.2 Indicadores de la Recuperación de Información

En la Recuperación de Información se han propuesto diferentes indicadores para medir cuantitativamente la performance de los sistemas de recuperación de información clásicos [Losee, 1998], la mayoría de los cuales pueden ser extendidos para evaluar la búsqueda en la web. Las medidas que se definen en un modelo básico de Recuperación de Información son Precisión y Recall.

2.4.2.1. Precisión

La *precisión* es la proporción de documentos recuperados relevantes, del total de los documentos recuperados. Es decir:

$$P = \frac{C_{DRR}}{C_{DR}}$$

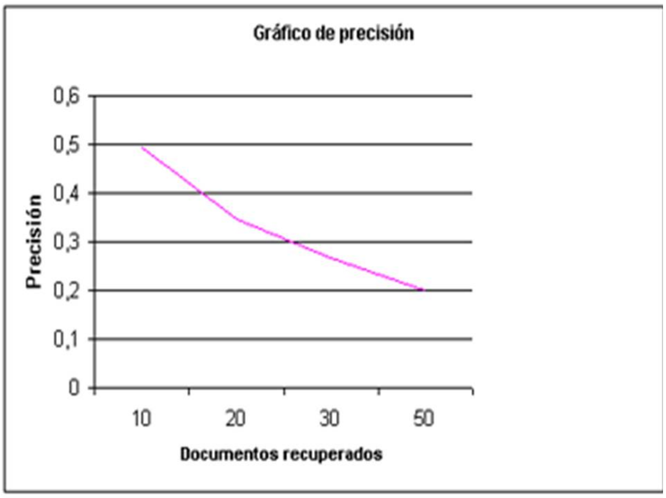
donde:
P es la Precisión
C_{DRR} es la cantidad o número de documentos relevantes recuperados
C_{DR} es la cantidad o número de documentos recuperados

El resultado de esta operación está entre 0 y 1. Así, la recuperación perfecta es en la que únicamente se recuperan los documentos relevantes y por lo tanto tiene un valor de 1.

Esta medida está relacionada con dos conceptos, el de ruido y el de silencio informativo. De este modo, cuanto más se acerque el valor de la precisión a 0, mayor será el número de documentos recuperados que no le sirvan al usuario y por lo tanto el ruido que encontrará será mayor.

La salida obtenida en la recuperación es ordenada en función de la relevancia, por lo que los documentos más relevantes están al comienzo, de esta manera a medida que avanzamos en el número de documentos recuperados, la precisión decae.

La figura siguiente muestra la representación gráfica de la precisión:



Los sistemas más precisos son aquellos que en su gráfica describen una curva con valores altos al principio y que van decreciendo.

2.4.2.2. Recall / Sensibilidad / Exhaustividad

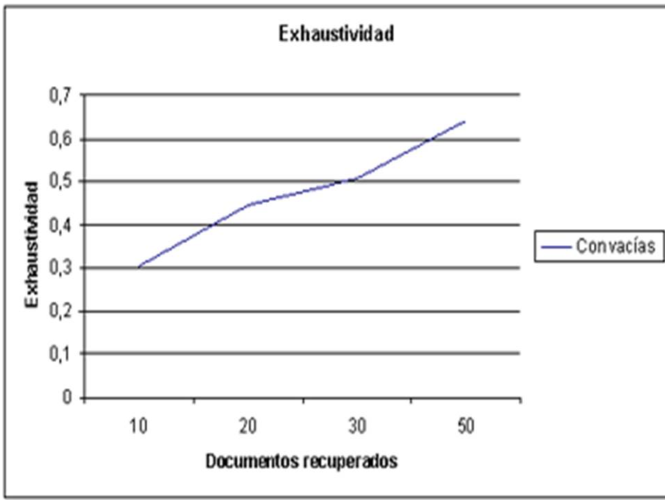
El *recall*, también conocido como sensibilidad o como exhaustividad, se define como la proporción de los documentos relevantes que son recuperados. Esto es:

$$R = \frac{C_{DRR}}{C_{TDR}}$$

donde:
R es el Recall
C_{DRR} es la cantidad o número de documentos relevantes recuperado
C_{TDR} es la cantidad o número total de documentos relevantes en la colección

es decir, es la proporción de material relevante recuperado, del total de los documentos que son relevantes en la base de datos, independientemente de que éstos, se recuperen o no. Esta medida es inversamente proporcional a la precisión. Si el resultado de este cálculo tiene como valor 1, tendremos la exhaustividad máxima, encontramos todo lo relevante que había, por lo tanto la recuperación es perfecta.

En la figura siguiente se presenta una representación gráfica de este indicador.



2.4.2.3. Ejemplo

Sea una base de datos que contiene 500 documentos y 50 corresponden a la definición del problema. El sistema recupera 75 documentos, pero solo 45 se refieren al problema. ¿Cuáles son los valores de recall y de precisión?:
Recall = 45 / 50 = 0.9 es decir el Recall es del 90 %
Precisión = 45 / 75 = 0.6 es decir la Precisión es del 60 %ç

Estos indicadores están inversamente relacionados. Es decir, cuando la Precisión aumenta, el Recall normalmente baja y viceversa. La precisión depende del nivel del usuario: los usuarios experimentados pueden trabajar con un Recall alto y una Precisión baja, porque son capaces de examinar la información y rechazar fácilmente la irrelevante. Los usuarios novatos, por otro lado, necesitan más alta Precisión porque les falta experiencia. Si la tolerancia para errar es alta y la tarea no es en tiempo crítico, esto puede ser aceptable para permitir al usuario revisar varios documentos hasta encontrar si uno es apropiado. De todos modos, si el tiempo es importante y el costo de cometer un error es alto, entonces la Precisión requerida es más alta. Además, cuando los términos son muy específicos aumenta la Precisión y baja el Recall. En cambio, cuando los términos son muy amplios aumenta el Recall y baja la Precisión. En general, un buen sistema de recuperación de información debe tratar de maximizar la recuperación de documentos relevantes, y minimizar la cantidad de los documentos irrelevantes recuperados.

Realizada una búsqueda en una colección de documentos, el conjunto de documentos recuperados no coincide totalmente con el conjunto de los relevantes sobre el tema de interés. Una búsqueda será óptima cuando estos dos conjuntos coincidan, es decir cuando todos los documentos recuperados sean relevantes y todos los documentos relevantes sean recuperados. Estos indicadores, que provienen de la Recuperación de Información tradicional se aplican a la Recuperación de Información en la Web.

2.4.3 Otros indicadores

La Precisión y el Recall son los indicadores más difundidos para evaluar los resultados de una consulta realizada. Además de estos existen diversos criterios para evaluar la eficacia y la eficiencia de la recuperación de información. El lector interesado puede consultar ampliaciones a este tema en [Baeza et al., 1999], [Martínez Méndez, 2004].

2.4.4 Colecciones de referencia

Existen colecciones estándares de prueba para la evaluación de un sistema de recuperación de información. El objetivo es permitir evaluar la efectividad de la recuperación de información

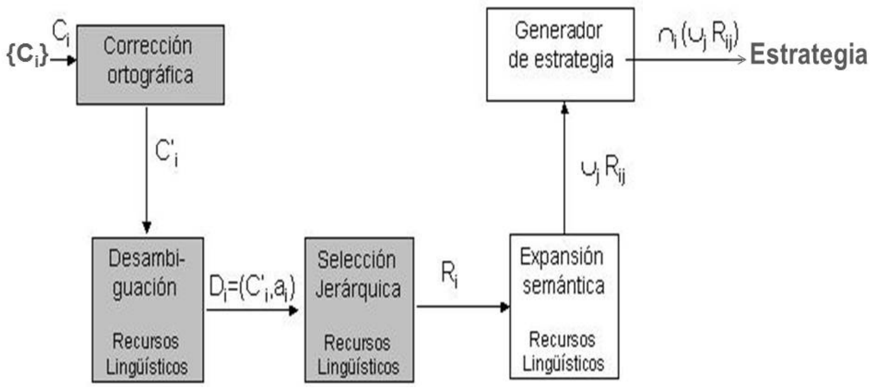
con medidas cuantitativas precisas. Entre ellas, la colección TREC (Text Retrieval Conference) promovida por el instituto NIST (National Institute of Standards and Technology) de Estados Unidos, y que contiene 1,89 millones de documentos, y evaluaciones de relevancia para 450 necesidades de información. Otras colecciones que se pueden mencionar son la colección Cranfield, de documentos recolectados en Inglaterra; NTCIR (NII Test Collections for IR Systems), proyecto que ha construido varias colecciones de prueba de medidas similares a las colecciones TREC; CLEF (Cross Language Evaluation Forum) que se concentra en idiomas europeos y en recuperación de información multilingual. Entre otros autores que tratan este tema, se puede consultar [Martínez Méndez, 2004] y [Baeza et al., 1999],

2.5 Estrategia de búsqueda

2.5.1 Expansión semántica de la consulta

Una estrategia de búsqueda es una expresión lógica compuesta por distintos conceptos combinados con los conectores lógicos de conjunción, disyunción y negación. La *expansión semántica* consiste en incorporar a la búsqueda términos que sean conceptualmente equivalentes: sinónimos y términos relacionados. Los sinónimos son grupos de palabras que representan un mismo concepto. Los términos relacionados son términos alternativos que, sin ser sinónimos ni estar relacionados jerárquicamente, pueden ser útiles para ampliar la cantidad de documentos a recuperar. Además, si el usuario desea obtener información en más de un idioma, entre estas expansiones se pueden incorporar la traducción de dichos términos. Esta expansión se puede realizar en forma automática mediante el uso de *recursoslingüísticos* adecuados.

Una arquitectura de la expansión semántica de una consulta se presenta en la figura siguiente:



Para realizar una consulta, el usuario ingresa un conjunto de conceptos {Ci} con $1 \leq i \leq n$ y la salida es una estrategia de búsqueda expandida semánticamente y asociada a estos conceptos.

Como se indica en la figura, para cada concepto Ci ingresado por el usuario, se obtiene el concepto C'i corregido ortográficamente; luego el concepto Di desambiguado; a continuación el concepto Ri jerárquicamente relacionado y finalmente, como resultado de la expansión semántica el conjunto unión de sinónimos y términos relacionados. En todas estas etapas se utilizan recursos lingüísticos.

Los conjuntos, formados por la unión de los Rik, ingresan al *Generador de estrategia* cuya salida es la intersección de estas uniones, con $1 \leq i \leq n$, donde n es la cantidad de los conceptos ingresados. La salida del Generador de estrategia contiene la estrategia de búsqueda asociada al interés del usuario.

Es decir, para la búsqueda que involucra los conceptos C1 y C2 y .... y (no Ch) y ... y Cn planteada por el usuario, se obtiene la estrategia siguiente:

(R11 OR R12 OR ... OR R1r)
AND
....
AND
(NOT (Rh1 OR Rh2 OR ... OR Rhr))
......
AND(Rn1 OR Rn2 OR ... OR Rnr)

donde:
(R11 OR R12 OR ... OR R1r) es la expansión del concepto C1
...
(NOT (Rh1 OR Rh2 OR ... OR Rhr)) es la negación de la expansión del concepto Ch
...
(Rn1 OR Rn2 OR ... OR Rnr) es la expansión del concepto Cn

y el valor de *r* depende de cada concepto, pues todos los conceptos pueden no tener la misma cantidad de expansiones. Esta estrategia también se puede representar en XML para un procesamiento posterior.

Por ejemplo, supongamos que el interés del usuario es saber la "relación de la aspirina en el tratamiento del cáncer de pulmón". Los conceptos que ingresa el usuario son: *cáncer de pulmón* - *aspirina* - *tratamiento*. La estrategia de búsqueda final provista por el Generador de estrategia es:

(lung neoplasms OR lung cancer OR cáncer de pulmón OR carcinoma of the lungs)
AND
(aspirina OR aspirin OR ácido acetil salicílico)
AND
(tratamiento OR treatment)

2.5.2 Recursos lingüísticos para la expansión

Los recursos lingüísticos descriptos en este apartado se utilizan como ayuda para la preparación de estrategias de búsqueda adecuadas que representen la necesidad de información del usuario. Esto es, se utilizan para desambiguar los conceptos en el caso de tener varias acepciones, para permitir la selección de conceptos jerárquicamente relacionados y para expandir semántica y multilingualmente cada concepto, y de esta forma mejorar la recuperación de información. Los recursos lingüísticos pueden ser de tipo general o especializado en algún dominio del conocimiento.

Los recursos lingüísticos no sólo pueden utilizarse para la preparación de la estrategia de búsqueda, sino también para la clasificación e integración de la información. En la *clasificación* de la información resultante de por ejemplo las páginas web obtenidas a través de un buscador, estos recursos permiten reconocer conceptos similares. Tener, por ejemplo, una ontología que agrupe los conceptos {'proyecto de investigación', 'trabajo de investigación', 'research project'}, ayudaría a clasificar en un mismo grupo, páginas que contengan datos de proyectos de investigación que se estén realizando sobre el tema buscado. En la *integración* de la información obtenida a partir de las distintas fuentes los recursos permiten unificar conceptos expresados con distinta terminología y reconocer coincidencias de autores o instituciones que puedan estar expresadas de distinta manera. Por ejemplo, reconocer que dos documentos provienen de una misma institución si en los respectivos documentos la Institución, contiene el valor "MIT" en uno de ellos y el valor "Massachusetts Institute of Technology" en el otro.

2.5.2.1. Diccionarios

Un diccionario indica las distintas acepciones de un término y permite su expansión con sinónimos. Algunos de los diccionarios permiten además la expansión con otros términos

relacionados jerárquica y/o semánticamente a cada acepción del término, como ser merónimos, hipónimos e hiperónimos. La *sinonimia* es la relación entre términos con un mismo significado. Por ejemplo, "cáncer" y "neoplasma". La *meronimia* es la relación semántica entre un término que denota una *parte* y el que denota el correspondiente *todo*. Por ejemplo, "brazo" es una parte (merónimo) del "cuerpo". La *hiponimia* es una relación de subordinación entre términos, es decir un término es un hipónimo de otro término si su significado está incluido en el del segundo. Por ejemplo, "gorrión" es un tipo (hipónimo) de "pájaro". La *hiperonimia* es una relación de superordenación entre términos, es decir un término es un hiperónimo de otro término si su significado incluye al del segundo. Por ejemplo, "animal" (hiperónimo) incluye al término "pájaro".

Un diccionario muy utilizado como recurso general es WordNet [Miller, 1995]. Puede ser descargado (www.cogsci.princeton.edu/{~}wn/) o se puede consultar en línea. El diccionario de la Real Academia Española se puede consultar en línea (www.rae.es).

2.5.2.2. Diccionarios multilinguales

Para aumentar el número de documentos a recuperar se puede ampliar cada concepto en los idiomas deseados por los usuarios mediante el uso de diccionarios multilinguales generales y especializados que permiten traducir un concepto a otros idiomas. Algunos diccionarios multilinguales están disponibles en la web. Por ejemplo, Eurowordnet es un diccionario multilingual con conceptos en los idiomas de la Comunidad Europea que se puede consultar en línea. Algunos otros diccionarios multilinguales son: Foreignword.com (www.foreignword.com/), Diccionarios.com (www.diccionario.com/), Online English to Spanish to English Dictionary (www.freedict.com), Babylon, Diccionario y Traductor (www.babylon.com/), y WordReference.com (wordreference.com/), entre otros.

Actividad para el alumno: Comparar los resultados de la traducción de una estrategia de búsqueda con cada uno de estos diccionarios. ¿Son sólo diccionarios o también traducen? Analizar si permiten traducir sólo palabras individuales, frases, textos largos. ¿Es posible integrarlos en una sistema automático de generación de una estrategia de búsqueda expandida? ¿Sólo se consultan en línea o tienen versiones para descargar?

2.5.2.3. Tesauros

Un tesauro es un instrumento de control terminológico utilizado para traducir a un lenguaje más estricto el idioma natural empleado en los documentos. Por su estructura, es un vocabulario controlado y dinámico de términos relacionados semántica y genéricamente, los cuales cubren un dominio específico del conocimiento. (UNESCO).

Un tesauro está estructurado formalmente para hacer explícitas las relaciones entre conceptos. Está constituido por términos organizados mediante relaciones entre ellos y provistos de notas de alcance o de definición de los conceptos. La estructura de la terminología de un tesauro está basada en las interrelaciones entre los conceptos. Estas interrelaciones pueden ser: jerárquicas, de afinidad, y preferenciales. Las relaciones jerárquicas indican términos más amplios o más específicos de cada concepto. Las relaciones de afinidad muestran términos relacionados conceptualmente, pero que no están ni jerárquica ni preferencialmente relacionados. Las relaciones preferenciales se utilizan para indicar cuál es el término preferido o descriptor entre un grupo de sinónimos; y para la calificación de homónimos para diferenciar su significado eligiendo un significado preferido para cada término.

En las bases de datos documentales se utilizan palabras claves para describir el contenido de un documento. Estas palabras claves, o descriptores, pueden estar formadas por un término o por una frase que se eligen de un diccionario de términos controlados o permitidos para el sistema, es decir, de un tesauro. Así, el tesauro representa una herramienta documental que

permite la conversión del lenguaje natural de un documento al lenguaje controlado documental. En el lenguaje natural, existen sinónimos y homónimos. Los sinónimos son grupos de palabras que representan el mismo concepto, por ejemplo agua y H2O. Los homónimos son palabras que representan más de un concepto, por ejemplo banco, que puede referirse al mueble o a la institución financiera. El control de vocabulario implica la selección de un término preferido, también conocido como descriptor o palabra clave, entre un grupo de sinónimos; y la calificación de homónimos eligiendo un significado preferido para cada término [Lancaster, 1995].

En un tesauro se utiliza el término reservado USE para indicar cuál es el término preferido en el caso de sinónimos. Por ejemplo para el término *drugaddiction* se indica que el término preferido es substance dependence(USE substance dependence). Los términos prohibidos se representan en letra cursiva. La relación inversa de USE es UF (Used For). Por ejemplo substance dependence es UF *drugaddiction y* UF *drugdependence.* Esto significa que si en un documento aparece la frase *drug addiction* se utilizará substance dependence como término descriptor de dicho documento. Para homónimos o para indicaciones de múltiples alternativas se utiliza SEE. Por ejemplo:

- *processing*
- SEE fabrication
- OR reprocessing

donde se indica que el término *processing* es prohibido porque representa más de un concepto. El tesauro indica cuáles serán los términos adecuados para cada significado. La relación inversa de SEE es SF (Seen For).

Para las relaciones jerárquicas se utiliza la sigla BT (*Broader Term*) para indicar conceptos más amplios, y la sigla NT (*Narrower Term*) para indicar un concepto más específico. Por ejemplo, en la siguiente jerarquía:

- LUNG NEOPLASMS
  - BT1 Neoplasms
  - NT1 Carcinoma Bronchogenic
  - NT1 Coin Lesion, Pulmonary
  - NT1 Pulmonary Blastoma

el concepto Neoplasms incluye (es un cóncepto más amplio) al concepto Lung Neoplasms y éste incluye a Pulmonary Blastoma, que es un tipo particular (es un concepto más específico) de cáncer de pulmón.

La sigla RT se utiliza para mostrar conceptos relacionados conceptualmente con carácter horizontal. Este tipo de relación se establece entre términos que no son sinónimos ni pueden relacionarse jerárquicamente, pero que permiten una asociación entre ellos. Por ejemplo, los grupos etarios tienen como términos relacionados Adolescentes, Adultos, etc.; Padre y Madre también son términos relacionados.

Los términos del tesauro se clasifican en *descriptores*, o términos principales o preferidos o permitidos; y *nodescriptores*, es decir, términos equivalentes de carácter secundario o no preferidos o prohibidos. Los términos no descriptores no pueden ser utilizados como palabras claves de los documentos para su indización ni como términos de búsqueda. Para cada término no descriptor, el tesauro indica cual es el término permitido correspondiente para representar dicho concepto.

La mayoría de los tesauros existentes están actualmente disponibles en línea. Por ejemplo, algunos tesauros especializados son: en el dominio de las bellas artes el Art & Architecture Thesaurus Browser (`www.getty.edu/research/tools/vocabulary/aat/index.html`); en biblioteconomía y documentación el tesauro Library of Congress Classification (`lcweb.loc.gov/catdir/cpso/lcco/lcco.html`); en el dominio de la biomedicina el Medical Subject Headings (MeSH) (`www.nlm.nih.gov/mesh/meshhome.html`); en las ciencias biológicas el

Life Sciences Thesaurus (www.csa.com/e_products/databases-collections.php); en física y astronomía el tesauro PACS (www.aip.org/pacs/); en el dominio de la ingeniería el NASA Thesaurus (www.sti.nasa.gov); en lengua y literatura el Merriam Webster Thesaurus (www.m-w.com/thesaurus.htm); entre otros.

Actividad para el alumno: Elegir un conjunto de términos de aplicación en uno más dominios del conocimiento, y elegir algunos tesauros. Analizar la estructura de los tesauros mencionados y verificar si cumplen con las estructuras descriptas para los tesauros. Comparar los resultados de la expansión de una estrategia de búsqueda con cada uno de los tesauros elegidos. ¿Es posible integrarlos en un sistema automático de generación de una estrategia de búsqueda expandida? ¿Sólo se consultan online o tienen versiones para descargar?

2.5.2.4. Ontologías

Las ontologías proporcionan una vía para representar el conocimiento y son un enfoque importante para capturar semántica. La definición más consolidada es la que la describe como "una especificación explícita y formal sobre una conceptualización compartida" ([Gruber, 1993], [Studer, 1998]). Es decir, las ontologías definen conceptos y relaciones de algún dominio, de forma compartida y consensuada; y esta conceptualización debe ser representada de una manera formal, legible y utilizable por las computadoras.

Las ontologías tienen los siguientes componentes que sirven para representar el conocimiento sobre un dominio. Los *conceptos* que son las ideas básicas que se intentan formalizar. Las *relaciones* representan la interacción y enlace entre los conceptos del dominio. Suelen formar la taxonomía del dominio. Por ejemplo: subclase-de, parte-de, conectado-a, etc. Las *funciones*son un tipo de relación donde se identifica un elemento mediante un cálculo que considera varios elementos de la ontología. Por ejemplo, pueden aparecer funciones como categorizar-clase, asignarfecha, etc. Las *instancias* se utilizan para representar objetos determinados de un concepto. Y los *axiomas* que son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Por ejemplo: "Si A y B son de la clase C, entonces A no es subclase de B", "Para todo A que cumpla la condición C1, A es B", etc. Los axiomas permiten inferir conocimiento que no esté indicado explícitamente en la taxonomía de conceptos.

Actividad para el alumno: Analizar qué diferencia/s hay entre estos recursos: un diccionario, un tesauro, una ontología. Sugerir cuál o cuáles de estos recursos serían más adecuados en el caso de realizar una consulta, según diversos escenarios, por ejemplo una búsqueda en un dominio general, una búsqueda en un dominio específico, una búsqueda donde interese recuperar documentos en varios idiomas, una búsqueda en una base de datos documental, etc.

2.5.3 Caso de estudio

La expansión semántica permite formular una estrategia de búsqueda a partir de los conceptos ingresados por el usuario, resolviendo muchos de los problemas que se presentan en esta formulación como ser el correcto uso de la disyunción y de la conjunción, el uso correcto de paréntesis, la inclusión de sinónimos y palabras con distintas formas de escritura, la utilización de términos específicos, el uso correcto de la negación y los errores de tecleo. Así, es posible aumentar la cantidad de documentos recuperados y la precisión de los resultados.

Esto ha sido probado experimentalmente en dominios generales y en dominios específicos. [Deco et al., 2005a], [Deco et al., 2005b]. Para las pruebas se realizaron varias consultas a partir de distintos intereses de búsqueda, utilizando la estrategia planteada directamente por el usuario, y luego la estrategia expandida semánticamente. A modo de ejemplo, se muestra una tabla con algunos resultados de estas experimentaciones donde se utilizó el buscador Yahoo! y un recurso lingüístico para las estrategias de búsqueda expandidas.

	Búsqueda en Yahoo!					Búsqueda con expansión semántica en Yahoo!						
Descripción	Expresión búsqueda	Cantidad docs. resultantes	Cant. docs. relevantes en los primeros docs.			Términos para el refinamiento semántico	Estrategia de búsqueda	Cantidad docs. resultantes	Cant. docs. relevantes en los primeros docs.			Nivel de usuario
			10	20	50				10	20	50	
Uso de lincomicina en embarazadas	Uso de lincomicina en el embarazo	94	3	3	5	Lincomicina, Embarazo	("Lincomicina" OR "Lincomycin") AND ("Embarazas" OR "Neophroparty")	103	3	8	14	Inexperto
Colitis Ulcerosa	"Colitis Ulcerosa"	493000	4	6	13	Colitis Ulcerosa	"Colitis Ulcerosa" OR "Colitis ulcerative" OR "Ulcerative colitis"	3640000	9	16	40	Medio
Sindrome de Sjögren	"Sindrome de Sjögren"	15200	3	6	12	Sindrome de Sjögren	"Sindrome de Sjogren" OR "Sjogren Syndrome" OR "Sjogrens Syndrome" OR "Syndrome Sjogren's" OR "Sicca Syndrome" OR "Syndrome, Sicca"	1030000	6	11	31	Medio
Polimialgia reumatica	Polimialgia reumatica	2010	4	6	11	Polimialgia reumatica	"Polimialgia reumatica" OR "Polymyalgia rheumatica"	322000	2	7	28	Inexperto
Enfermedad de Munchausen	"Enfermedad de Munchausen" OR "Sindrome de Munchausen"	608	5	7	7	Sindrome de Munchausen	"Sindrome de Munchausen" OR "Syndrome Munchausen" OR "Syndrome Hospital-Addiction" OR "Munchhausen Syndrome" OR "Hospital-Addiction Syndrome"	20700	5	7	17	Medio
Colotis Pseudomembranosa	Colotis Pseudomembranosa	1070	4	8	18	Colotis Pseudomembranosa	"Pseudomembranous Colitis" OR "Colitis pseudomembranous" OR "Pseudomembranous colitis"	121000	8	17	27	Inexperto
Tratamiento con insulina en la embarazada diabética	"Tratamiento" "Insulina" "Embarazadas Diabéticas"	74	4	8	15	Tratamiento, Insulina, Embarazadas Diabéticas	("Tratamiento" AND "Insulina" AND "Embarazadas Diabéticas") OR ("Treatment" AND "Insulin" AND "Neophropathy Diabetic")	58	4	6	18	Experto
Cómo evoluciona el tratamiento del HPV con una cirugía	Tratamientos HPV	69400	8	14	34	"HPV" and (cricirugia or crioterapia or electrocoagulación)	HPV AND (cricirugía OR crioterapia OR electrocoagulación)	19	9	16	16	Medio
Vacuna contra el cáncer de piel	"skin cancer vaccine"	2030000	10	12	19	"skin cancer vaccine"	(vacuna "cáncer piel") OR ("skin cancer" vaccine)	1130	9	16	26	Experto
Diferentes manifestaciones de hipotiroidismo en la niñez	hipotiroidismo en la niñez	704	9	15	35	hipotiroidismo, niñez	hipotiroidismo AND (childhood OR niñez)	10700	10	18	42	Medio
Infarto agudo de miocardio	infarto agudo miocardio	407000	8	16	33	infarto, agudo, miocardio	"infarto agudo miocardio" OR (infarct, myocardial) OR (infarction, Miocardial)	58600	8	17	42	Medio
Úlcera gástrica	úlcera gástrica	315000	8	14	24	úlcera, gástrica	("úlcera gástrica") OR ("gastric ulcer") OR ("stomach ulcer")	30300	10	18	38	Inexperto

Actividad para el alumno: Elegir un tema de su interés y un buscador. Generar la lista de términos que representan el o los conceptos de interés. Evaluar los resultados obtenidos a partir de diversas estrategias de búsqueda teniendo en cuenta el uso de nombres propios y siglas, frases, errores ortográficos, términos con pocos o ningún sinónimo, términos más específicos, sinónimos y siglas polisémicas agregados del recurso lingüísico elegido para la expansión. Evaluar la precisión utilizando los primeros 50 documentos. ¿Se puede evaluar el recall? ¿Por qué? ¿Qué tipo de recurso es recomendable para una búsqueda en un dominio específico, por ejemplo medicina?

2.6 Indexado y búsqueda

Dado un cuerpo de documentos, las tareas de un sistema de recuperación de información son la indexación, la búsqueda y la visualización de dichos documentos. La tarea de indexación de documentos significa construir archivos de acceso a éstos. Esta tarea produce un archivo, que es llamado *índice invertido*, que se construye a partir del cuerpo de documentos, realizando un mapeo entre cada palabra presente en el cuerpo hacia los documentos que la contienen [Baeza et al., 1999], [van Rijsbergen, 1979].

Los índices guardan una estrecha relación con la forma de representación elegida para los documentos. La construcción de índices puede hacerse por palabras, frases u oraciones enteras. Si los documentos se representan como una bolsa de palabras, considerando como *keywords* la totalidad de sus palabras, entonces los índices estarán construidos para buscar palabras. Dado que el reconocimiento de frases no mejora apreciablemente el proceso de recuperación y es un proceso no trivial, el método más común de construcción de índices es por palabras. Una búsqueda de frases puede resolverse usando índices construidos por palabras, si bien la operación resulta más costosa.

2.6.1 Preprocesamiento de los documentos

A los efectos de construir un índice, es necesario generar una lista de palabras a partir de la colección o cuerpo de documentos. Para identificar y extraer automáticamente las palabras de un texto se utiliza un proceso llamado tokenización (o *tokenization*). Un tokenizador reconoce automáticamente los límites de las palabras. Este trabajo no es tan trivial como puede parecer, ya que se presentan dificultades. Por ejemplo, ante siglas ("U.N.R." debe ser considerado un token y no tres), o palabras compuestas (como las separadas por guiones: en "producción argentino-española" hay tres tokens, en cambio en "ciudadano italo-americano" hay dos). Este reconocimiento de unidades sintácticas es simple en idiomas como el inglés o el español, pero se puede volver complicado en idiomas con palabras compuestas no separadas por espacios como el alemán y puede ser imposible en lenguas como el chino o el árabe.

Una vez extraídas las palabras de un texto, se puede realizar un preproceso sobre el conjunto resultante antes de construir los índices para reducir el tamaño de los mismos, reduciendo el número de términos a través de los procesos de *stemming* y de *lemmatization* . El proceso de *stemming* consiste en obtener la raíz de las palabras, de forma que el proceso de búsqueda se realice sobre las raíces y no sobre las palabras originales. Este proceso es un conjunto de reglas para remover sufijos dejando sólo las raíces sintácticas de los términos, reemplazando plurales y verbos regulares conjugados por singulares e infinitivos respectivamente. Esto es reemplazar las variantes morfológicas de las palabras por sus raíces. Así, bajo la suposición de que dos palabras que tengan la misma raíz representan el mismo concepto, el sistema de recuperación de información podrá relacionar términos presentes en la consulta con los que se encuentren en los documentos y que aparezcan en alguna de sus variantes morfológicas. Los primeros algoritmos de *stemming* se desarrollaron para el idioma inglés. Pero esta técnica necesita ser adaptada para lenguas que presentan características distintas al inglés, como ser idiomas más flexivos, tal como el español. Uno de los algoritmos más utilizados para el inglés, es el de Porter [Porter, 1980]. También existen algoritmos para otras lenguas tales como el francés [Savoy, 1999], el español [Figuerola et al., 2002], el holandés [Kraaij & Pohlmann, 1994], el griego [Kalamboukis, 1995] y el latín [Schinke et al., 1996]. En general, estos algoritmos se basan en un conjunto sencillo de reglas que truncan las palabras hasta obtener una raíz común. En el proceso de *lemmatization* en lugar de usar algoritmos para remover partes de las palabras, se utiliza un diccionario para reemplazarla completamente por su raíz sintáctica. Se emplea por ejemplo para dejar en infinitivo los verbos irregulares conjugados. Estos procesos presentan como dificultad que pueden llevar a inconsistencias, para citar un caso: "cupo" puede reducir a "caber", lo que es correcto si la palabra "cupo" representaba a este verbo conjugado, pero no si se refería al sustantivo "lugar". Estas dos reglas son totalmente dependientes del lenguaje de los documentos y son difíciles de aplicar en contextos como la Web en el que coexisten documentos en diferentes de idiomas. Otra técnica para disminuir el tamaño de los índices es la remoción de palabras muy frecuentes que resultan irrelevantes o no significativas para resolver cualquier consulta, como artículos, preposiciones y conjunciones. Estas palabras reciben el nombre de *stopwords*. El hecho de considerar a una palabra como un *stopword* depende en parte del contexto en que aparezca. Los términos que aparecen en todos los documentos de una colección, son inútiles para discriminar un documento de otro, y pueden ser considerados *stopwords*. Por ejemplo, en una colección de textos sobre informática, las palabras "computadora", "PC" y "ordenador" serían *stopwords*.

Realizados estos procesos, se obtiene un vocabulario del cuerpo de documentos. Con este vocabulario trabaja el sistema para generar el archivo invertido.

2.6.2 Archivo Invertido

Se describe a continuación la implementación de índices en la forma de archivos invertidos, por ser el mecanismo de implementación de índices más utilizado y por ser directamente aplicable

al modelo Espacio Vectorial.

Los archivos invertidos (*inverted files*) son un mecanismo orientado a palabras para indexar una colección de documentos. Un archivo invertido se compone de dos partes: el vocabulario y las listas de ocurrencias (*posting list*).

El vocabulario puede estructurarse de diferentes formas. Por ejemplo, los términos pueden estar ordenados secuencialmente, mediante árboles o tablas *hash*. Esto afectará la forma en que se realicen las búsquedas de términos dentro del vocabulario.

Las listas de ocurrencia almacenan las ocurrencias de cada término del vocabulario, y una lista de referencias (o punteros) a todos los documentos en los que aparece dicho término. Además, pueden contener información sobre la posición de las palabras en el texto, que puede estar medida en palabras o en caracteres, la frecuencia de aparición de las mismas, sus pesos, etc. La posición de las palabras en el texto se utiliza para realizar búsquedas por adyacencia. El peso se utiliza para un posterior ranking de importancia.

Por ejemplo, sea el siguiente texto perteneciente a un documento:

Esto es un texto. Un texto tiene muchas palabras. Las palabras están compuestas de letras.

Utilizando la técnica de tokenización indicada en el apartado 2.6.1. se obtiene el vocabulario siguiente con la siguiente información adicional: la posición del primer caracter de la palabra y el número de palabra dentro del párrafo. El vocabulario está ordenado alfabéticamente.

Vocabulario	Posición del primer caracter	Número de palabra
Compuestas	70 ...	13 ...
De	81 ...	14 ...
Es	6 ...	2 ...
Están	64 ...	12 ...
Esto	1 ...	1 ...
Las	51 ...	10 ...
Letras	84 ...	15 ...
Muchas	34 ...	8 ...
Palabras	41, 55 ...	9, 11 ....
Texto	12, 22 ...	4, 6 ...
Tiene	28 ...	7 ...
Un	9, 19 ...	3, 5 ...

A este archivo se agrega la información del documento de donde se obtuvo, se realiza el *stemming* y/o la *lemmatización* y se eliminan las palabras no significativas (*stopwords*) y se genera así el archivo invertido.

El proceso de búsqueda en los archivos invertidos está dividido en tres partes. Primero se buscan las palabras de la consulta en el vocabulario. Esto puede hacerse por medio de una búsqueda secuencial, binaria, con árboles, tablas hash, etc., dependiendo de cómo se haya estructurado al vocabulario. Si la consulta está compuesta por varias palabras, como por ejemplo una frase, se la divide en los términos que la constituyen y se procede de esta forma separadamente para cada uno de ellos. Luego se recuperan los documentos en que ocurren dichas palabras, simplemente recorriendo las listas de ocurrencias de esas palabras. Si la consulta estaba compuesta por una sola palabra, el proceso termina aquí y se muestran los resultados, que pueden estar ordenados por un ranking de relevancia. Si la consulta estaba compuesta por varias palabras, en la tercera

fase se termina de resolver la consulta procesando los datos obtenidos. Si la consulta es una frase se busca entre los documentos obtenidos en el paso dos, aquellos en los que las palabras que componen la frase aparezcan en forma consecutiva. Esta verificación puede realizarse utilizando la información sobre las posiciones de las palabras dentro de cada documento. Si todas las palabras de la frase aparecen en un dado documento, se verifica que sean consecutivas y que respeten el orden de la frase buscada. Si la consulta contiene varias palabras se buscarán los documentos que contengan al menos a una de ellas y se le otorgará un ranking mayor a aquellos en los que figuren más términos de la consulta y a aquellos en los que las palabras ocurran en posiciones próximas, evitando así retornar textos extensos donde es posible que figuren las palabras en varias partes pero en distintos contextos.

En el caso de que en la consulta aparezcan los operados booleanos AND, OR, o NOT se realizará la intersección, unión, complemento o diferencia, respectivamente, de los conjuntos de documentos retornados para cada término.

2.6.3 Ventajas y desventajas del Archivo Invertido

Los archivos invertidos son el tipo de índices más utilizado en la actualidad y poseen un formalismo simple y eficiente. Dentro de sus principales ventajas se tienen la capacidad de manejar pesos no binarios y así poder medir la similitud entre un documento y una consulta en forma gradual, a diferencia del modelo booleano. Además la medida de similitud usada proporciona un ranking que permite ordenar los documentos resultantes de acuerdo a su relevancia.

Algunas desventajas de los archivos invertidos son que asumen a los textos como secuencias de palabras, limitando las búsquedas que pueden realizarse, y que algunas consultas complejas, como las consultas de frases, pueden ser costosas de resolver, conviniendo en este caso el empleo de otras técnicas tal como vectores de sufijos.

En los vectores de sufijos, es posible buscar prefijos, palabras y frases directamente en los árboles de sufijos. La búsqueda se realiza comparando la cadena buscada con las entradas hasta hallar una que la contenga como subcadena en su inicio. Así, la búsqueda de frases no es más compleja que la búsqueda de palabras en los vectores de sufijos. La ventaja principal que ofrecen los vectores de sufijos es que permiten resolver de manera más eficiente algunas consultas complejas, como en el caso de la búsqueda de frases. Un inconveniente de esta estructura de índices es que su construcción es un proceso costoso.

En general, para aplicaciones que utilizan exclusivamente palabras, los archivos invertidos son más eficientes, excepto para la resolución de consultas complejas.

2.6.4 En la web

Todo lo descripto en este apartado se extiende a los buscadores web, con la consideración de que cada página es el equivalente a un documento. Aquí se puede mencionar que en un ranking de las respuestas los algoritmos tienen en cuenta también información de los enlaces, pues un enlace representa una relación entre páginas. Estos enlaces se utilizan para dar peso a una página. Puede mencionarse el algoritmo PageRank, utilizado por Google, que fue diseñado por integrantes de la Universidad de Stanford [Page et al., 1998]. Otro algoritmo de ranking de resultados es HITS (Hyperlink Induced Topic Search) [Kleinberg, 1998] [Kleinberg, 1999]. Una diferencia entre PageRank y HITS es que el primero se calcula para todas las páginas web recopiladas por el motor y almacenadas en su base de datos, previo a la realización de consultas. En cambio HITS se ejecuta sobre el conjunto de páginas web recuperadas para cada consulta, en tiempo real. Otra diferencia es que HITS se basa en el cálculo de confiabilidad y centralización, en cambio PageRank se basa sólo en el cálculo de confiabilidad.

Actividad para el alumno: Realizar un relevamiento de técnicas utilizadas por los sistemas

de recuperación de información para construir el vocabulario. Realizar un relevamiento de las estructuras de índices utilizadas, incluyendo estructuras de datos adicionales que aporten a la velocidad en la respuesta ante una consulta.

2.7 Algunas aplicaciones

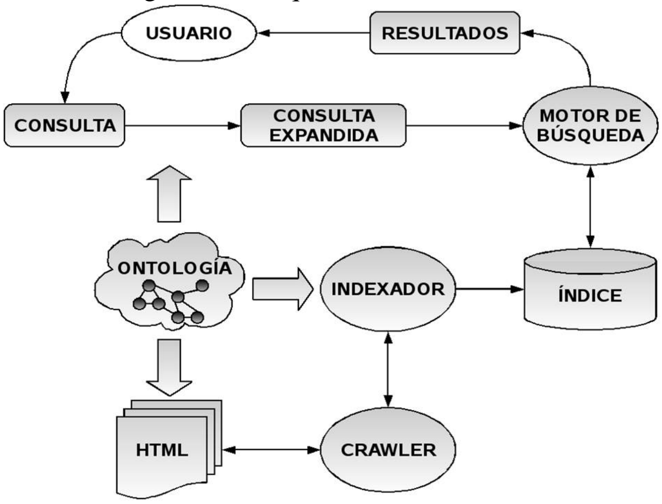
2.7.1 Consultas en un sitio web

En la actualidad, existen numerosos sitios web que manejan un caudal de información considerable. Estos sitios suelen tener un buscador propio para permitirle al usuario localizar y acceder a la información que necesita en una forma más directa y más rápida, evitando tener que navegar todo el sitio. Al igual que la búsqueda en la web, en la búsqueda dentro de un determinado sitio, muchas veces cuando se busca información a través de un buscador, no se obtienen los resultados esperados. Estas búsquedas infructuosas pueden ocurrir debido a un conjunto de factores que inciden negativamente en los resultados de las búsquedas. En este tipo de escenarios se presentan también problemas de sinonimia y polisemia de términos.

Un buscador tiene tres componentes principales: un *crawler,* un indexador y un motor de búsqueda. El *crawler* recorre la colección y crea un repositorio local con el contenido de las páginas visitadas. El indexador procesa dicho repositorio y genera un índice invertido del mismo. Las consultas ingresadas son enviadas al motor de búsqueda. Este último consulta al índice y retorna los resultados de las búsquedas al usuario.

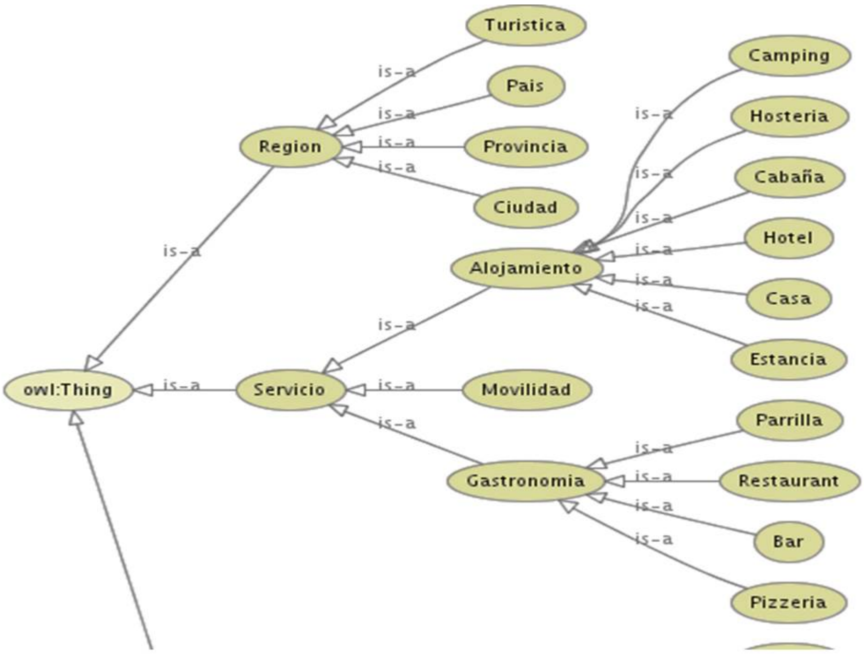
En la figura 2.1 se muestra la arquitectura del buscador compuesta por un motor de búsqueda web (buscador), por ejemplo Nutch (`lucene.apache.org/nutch/`), un índice invertido, una ontología de dominio y una colección de páginas web etiquetadas con metadatos.

Figura 2.1: Arquitectura del buscador



La ontología de dominio se utiliza para definir un vocabulario controlado, con el fin de reducir la sinonimia y polisemia de términos de manera de etiquetar las páginas y así insertar información adicional (metadatos) en los documentos de la colección de modo que queden caracterizados mediante algún criterio y por lo tanto sean más fáciles de recuperar. Para tratar la sinonimia de términos el vocabulario controlado se construye de modo que sus términos sean los nombres de las clases de la ontología, ya que éstos designan los conceptos del dominio. Por ejemplo, para un dominio Turismo, dada una clase Alojamiento, que determina el término del vocabulario controlado *alojamiento*, se podría anotar al documento con las palabras: albergue, hospedaje y aposento, como sinónimos de alojamiento.

Para definir una ontología se utiliza un editor de ontologías, por ejemplo el editor Protègé-OWL (protege.stanford.edu). A modo de ejemplo, una pequeña parte de una taxonomía del dominio Turismo, se muestra en la figura siguiente.



El uso de una arquitectura de este tipo, donde se incorpora una ontología como recurso lingüístico propio de un sitio web, con el objetivo de tener un vocabulario controlado propio del dominio, y su utilización en el momento de indexación de las páginas del sitio y en el momento de la búsqueda para expandir la estrategia de búsqueda, mejora la precisión y la cantidad de documentos recuperados. Una ampliación de este tema puede consultarse en [Ponce et al., 2008].

Actividad para el alumno: Analizar alternativas a la arquitectura presentada en este apartado. Elegir un dominio de aplicación y proponer un diseño de una ontología de dominio para mejorar la indexación y/o la búsqueda en un sitio web. Evaluar el uso de otros recursos. ¿Sería conveniente utilizar más de un recurso? Justificar.

2.7.2 Búsquedas Inteligentes, Sistemas Recomendadores

En la búsqueda de información, en particular en las búsquedas en la web, se advierte una sobrecarga de información que obliga a los usuarios a explorar espacios excesivamente densos, convirtiendo la selección de la información que les interesa en una tarea tediosa, que insume mucho tiempo y que es difícil de realizar. Por otro lado, un material dado no es el adecuado para todos los usuarios, dado que los usuarios poseen características y preferencias personales, que deberían ser consideradas en el momento de la búsqueda. Un Sistema Recomendador ([Resnick et al., 1997], [Terveen et al., 2001]) ayuda a resolver este tipo de problema puesto que son capaces de seleccionar, de forma automática y personalizada, el material que mejor se adapte a las preferencias o necesidades de un usuario. Estos sistemas utilizan distintas técnicas para razonar sobre las preferencias de los usuarios (modeladas en perfiles personales) y sobre las descripciones semánticas del material disponible.

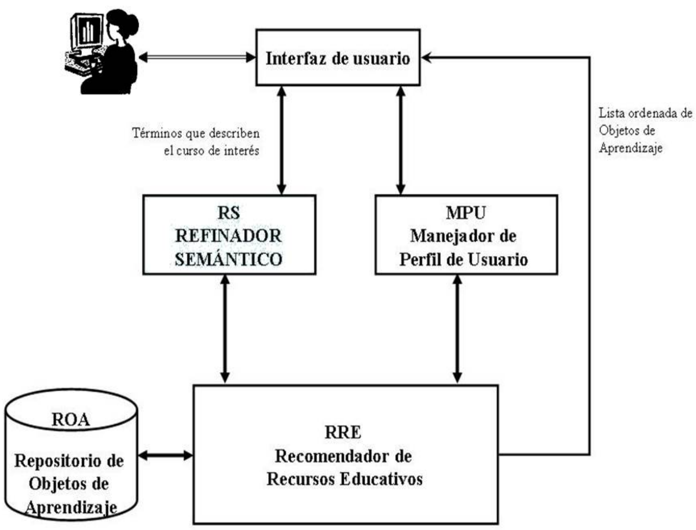
Este tipo de sistemas generalmente se modelan como sistemas multiagentes. Los agentes surgen dentro del campo de la Inteligencia Artificial y representan una nueva forma de analizar, diseñar e implementar sistemas de software complejos [Jennings et al. 1998]. Se puede definir un agente como una aplicación informática con capacidad para decidir cómo actuar para alcanzar

sus objetivos. Un agente inteligente puede funcionar fiablemente en un entorno rápidamente cambiante e impredecible, como es la web. Pueden configurarse con diferentes perfiles para tomar decisiones de acuerdo a las necesidades del usuario y hacer tareas más específicas y personalizadas.

La personalización de los resultados de una búsqueda se sustenta en los metadatos, tanto del usuario como de los documentos, dado que permiten evaluar la cercanía de un documento al perfíl del usuario. Los metadatos descriptivos de un documento, en general, siguen un estándar de metadatos; por ejemplo, para recursos educativos, el estándar utilizado es LOM (Learning Object Metadata) [IEEE].

A modo de ejemplo, una arquitectura para la recuperación de recursos educativos que le brinde al usuario el material que responda a su necesidad temática y a sus preferencias se presenta en la figura 2.2.

Figura 2.2: Arquitectura para la recuperación de Recursos Educativos



Se asume que existe un Repositorio de Objetos de Aprendizaje (ROA) con recursos educativos enriquecidos con metadatos que describen las características del objeto como ser el tema que cubre, su idioma, cantidad de imágenes, etc. Así, el módulo RS produce la estrategia de búsqueda asociada al interés del usuario y provee al módulo RRE el conjunto de documentos que satisfacen la búsqueda temática junto con sus metadatos. El módulo MPU obtiene y administra los perfiles de los usuarios. El módulo RRE evalúa la similitud entre las características de los documentos que satisfacen la búsqueda temática y el perfil del usuario. El resultado es una lista ordenada donde el primer elemento es el más cercano a su perfil. El módulo RRE puede diseñarse con tecnología de agentes.

Los metadatos pueden representarse, por ejemplo en la forma de vectores donde cada elemento del vector está compuesto por tres partes: el nombre del metadato, el valor que tiene el metadato y un peso que indique la importancia que posee esa característica para que el recurso sea aprovechado por el usuario.

La ventaja de un sistema recomendador es que permite mejorar los resultados de una búsqueda porque presenta los resultados de distinta forma según el usuario que haya realizado la consulta. Sin embargo, todavía uno de los mayores problemas encontrados para el uso de los sistemas recomendadores es la falta de metadatos en los documentos.

Ampliaciones y extensiones de este tema pueden encontrarse en [Bender et al, 2006], [Casali

et al, 2006] y en [Deco et al., 2008].

Actividad para el alumno: Analizar alternativas a la arquitectura presentada en este apartado. Elegir un dominio de aplicación. Evaluar el uso de estándares para definir los metadatos. Evaluar, frente al target de posibles usuarios, los metadatos de mayor relevancia para producir una recomendación. Proponer una arquitectura para un sistema recomendador. Justificar.

2.7.3 Recuperación de información en la web

En la tabla siguiente se presenta un cuadro comparativo de algunos proyectos que están relacionados con la recuperación de información en la web.

Proyecto	Descripción	Formalismo	Recursos utilizados	Enfoque de agentes	Refinamiento	Tipo de documento sobre el que actúa
CiteSeer	ACI orientado a buscar, indexar y recuperar papers	Modelo espacio vectorial Similitud de docs	Buscadores	Agentes	Expande la consulta a partir de las citas bibliográficas	Postscript, PDF
InfoSleuth	Recupera e integra información de fuentes heterogéneas		Ontologías	Red de agentes cooperantes	Utiliza las ontologías para mapear y para integrar, pero no para refinar la consulta	html
OntoBroker On2broker	Busca información e infiere respuestas en bases de datos cuya información es cargada a partir de páginas web	Frame Logic	Ontologías	La versión 2 utiliza agentes.	No realiza refinamiento semántico. Realiza inferencia sobre los datos	Bases de datos con información de págs web
OntoSeek	SRI basado en contenido.	LCG (grafos conceptuales léxicos). Compara grafos isomorfos	Ontologías	No	Desambiguación léxica y validación semántica.	Html y xml. Catálogos de productos y págs. amarillas on line.
Untangle	Recupera información de la web pero no con técnicas de IR, sino con técnicas de KR.	KR&R	Ontologías	No	No realiza refinamiento semántico. Utiliza ontologías para representar la estructura de los docs de la web	Html. Inicialmente para emails
WebFind	Descubre papers disponibles en la web por sus autores, en tiempo real	Similitud de docs	Bases de datos y fuentes de información externas: MelVyl, NetFind	No	Realiza la consulta detectando los autores de los documentos buscados y la amplía buscando papers de dichos autores relacionados con el tema.	Html. Páginas personales de los autores.
WebMate	SRI que asiste en la navegación de la web con perfiles personalizados de usuario.	Modelo espacio vectorial	Buscadores	Agentes Proxy: monitorea y aprende de las acciones del usuario Controlador applet: interactúa con el usuario	Expande la consulta utilizando términos obtenidos de un feedback de relevancia	Html

Actividad para el alumno: Indicar si los proyectos mencionados siguen activos, y en caso afirmativo en qué estado se encuentran. Realizar una búsqueda sobre proyectos vinculados a la recuperación de información que no estén en la tabla anterior. Proponer características adicionales a las dadas para comparar proyectos de este tipo.

2.7.4 Búsqueda de información multilingüe

La búsqueda de información multilingüe trata el problema de encontrar documentos que están escritos en otros idiomas, distintos al idioma de la consulta. Si se desea recuperar documentos en otro idioma, es necesario efectuar una traducción de la consulta para realizar la búsqueda en dicho idioma. Este proceso no es simple debido a la complejidad semántica del vocabulario.

El problema en una búsqueda multilingüe es que los idiomas de la consulta y de los documentos son distintos. Por lo tanto, es necesario efectuar una traducción para poder realizar una búsqueda en la que tanto la consulta como los documentos se encuentren en el mismo idioma. En el problema de la búsqueda de información multilingüe, la traducción de la consulta es la opción

más frecuente, porque su costo computacional es menor al costo de traducir los documentos. Los tres problemas principales para automatizar la traducción de la consulta, según [Grefenstette, 1998], son: saber cómo un término escrito en un idioma puede ser expresado en otro idioma; decidir cuáles de las posibles traducciones de cada término son las adecuadas en un contexto dado; y saber cómo medir la importancia de las diferentes traducciones que se consideran adecuadas. Estos problemas son compartidos por los sistemas de traducción automática y los sistemas de recuperación de información multilingüe.

En la búsqueda de información multilingüe el uso de recursos multilingües, tales como diccionarios multilingües y tesauros multilingües, en la expansión de la consulta permite realizar una expansión multilingüe de la consulta. Un ejemplo de diccionario multilingüe general es EuroWordNet [Vossen, 1998]. Un *tesauro multilingüe* sobre un área del conocimiento permite la traducción de términos específicos de ese dominio que quizá no puedan encontrarse en un diccionario. Un ejemplo de este tipo de tesauro sobre el dominio médico es UMLS (Unified Medical Language System), que es el Sistema Unificado de Terminología Médica de la Biblioteca Nacional de Medicina de Estados Unidos (National Library of Medicine).

Actividad para el alumno: Evaluar algunos diccionarios multilingües, disponibles en línea, para las traducciones entre los idiomas español, inglés y francés. Algunos recursos disponibles en línea que pueden utilizarse, pero no los únicos, son Systran (tr.voila.fr), Reverso (`www.elmundo.es/traductor/`), el servicio de SDL internacional (www.freetranslation.com/), Wordlingo (`www.worldlingo.com/en/products_services/worldlingo_translator`).
Para la evaluación de estos recursos utilizar un mismo conjunto de términos. Analizar qué sucede con la traducción en cada recurso; por ejemplo, evaluar si el recurso reconoce si el término es un sustantivo, un verbo (en infinitivo o conjugado) y lo traduce conservando esta categoría gramatical. Evaluar además, si la traducción es o no bidireccional.
Realizar una búsqueda de proyectos de investigación y aplicaciones que aporten a mejorar la efectividad en búsquedas multilingües. Proponer alguna solución que permita mejorar las propuestas encontradas. Evaluar la incidencia de la interacción con el usuario en estas propuestas.

2.7.5 Otros Problemas para Pensar

Algunos problemas adicionales propuestos son los siguientes:

- *Preparación para contingencias*. Esto es, cómo aumentar la cantidad de documentos recuperados si no se recupera información suficiente, y cómo reducir la cantidad si se recuperan demasiados documentos.
- *Selección automática del recurso lingüístico adecuado*, por ejemplo, utilizando un perfil de usuario realizar la selección automática de los recursos lingüísticos más adecuados para la generación de la estrategia de búsqueda.
- *Extracción automática de conceptos para la estrategia de búsqueda.* Proponer cómo generar una estrategia de búsqueda a partir del ingreso de una consulta en la forma de una frase escrita en lenguaje natural.
- *Utilización de ontologías con axiomas*. Esto permitiría poder realizar inferencias, y así poder incorporar a la estrategia de búsqueda otros conceptos además de los obtenidos desde los recursos lingüísticos.
- *Utilización de Feedback de Relevancia*. Evaluar el agregado a la estrategia de búsqueda de términos extraídos de documentos recuperados que el usuario indique que son relevantes para él. Analizar cómo y qué términos extraer de estos documentos señalados como relevantes.

2.8 Bibliografía

[Baeza et al., 1999] Baeza-Yates, R., Ribeiro-Neto, B. (eds.), Modern Information Retrieval. New York. ACM Press, 1999.
[Bender et al., 2006] Bender, C., Deco, C., Casali, A., Motz R. Una plataforma multiagente para la búsqueda de recursos educacionales considerando aspectos culturales. En Revista TE&ET (Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología). Vol. 1 Nro. 1. Editorial Responsable Red de Universidades Nacionales con Carreras de Informática (RedUNCI). ISSN 1850-9959. pp 20-29. 2006.
[Blair, 1990: 2-4] Blair, D. C. Language and Representation in Information Retrieval. Elsevier Science Publishers, 1990.
[Casali et al., 2006] Casali, A., Deco, C., Bender, C. and Motz, R. A multiagent approach to educational resources retrieval. En Proceedings Workshop on Artificial Intelligence for Education (WAIFE), 35ºJornadas Argentinas de Informática e Investigación Operativa. ISSN 1850 2784. pp 35-41. Mendoza, Argentina. 2006.
[Deco et al., 2005a] Deco, C., Bender, C., Saer, J., Chiari, M., Motz, R. Semantic refinement for web information retrieval. En Proceedings of the 3rd Latin American Web Congress. IEEE Press. pp 106-110, 2005.
[Deco et al., 2005b] C. Deco, C. Bender, J. Saer y M. Chiari. Expansión de consultas utilizando recursos lingüísticos para mejorar la recuperación de información en la web. En Víctor M. Castel, Comp. (2005) Desarrollo, implementación y utilización de modelos para el procesamiento automático de textos. Mendoza: Editorial de la Facultad de Filosofía y Letras, UNCuyo: 35-46. ISBN del soporte Internet: 987-575-019-0
[Deco et al., 2008] C. Deco, C. Bender, A. Casali, R. Motz. Design of a Recommender Educational System. En Proceedings 3ra. Conferencia Latinoamericana de Objetos de Aprendizaje LACLO 2008. Trabajo premiado entre los cinco mejores del Congreso. México. 2008.
[Figuerola et al., 2002] Figuerola, C. G., Gomez, R., Rodríguez, A. F. Z., Berrocal, J. L. A. Spanish Monolingual Track: The Impact of Stemming on Retrieval. In Peters, C., Braschler, M., Gonzalo, J., and Kluck, M., editors, Evaluation of Cross-Language Information Retrieval Systems, CLEF 2001, volume 2406 of LNCS, pages 253–261. Springer, 2002.
[Grefenstette, 1998] Grefenstette, G. The problem of CrossLanguage Information Retrieval, chapter in Cross-Language Information Retrieval. Kluwer Academic Publishers. 1998.
[Gruber, 1993] Gruber T. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Technical Report KSL-93-04, Knowledge Systems Laboratory, Stanford University, CA, 1993.
[IEEE] IEEE Learning Technology Standards Committe. LOM specification. Disponible en ltsc.ieee.org/wg12
[Kalamboukis, 1995] Kalamboukis, T. Suffix stripping with modern Greek. Program, 29:313–321, 1995.
[Kleinberg, 1998] Kleinberg J., Authoritative Sources in a Hyperlinked Environment. Proc. 9th Ann. ACM-SIAM Symp. Discrete Algorithms, ACM Press, New York, pp.668-677, 1998.
[Kleinberg, 1999] Kleinberg J.M., Authoritative Sources in a Hyperlinked Environment. Journal of the ACM, 46(5):604-632, 1999.
[Kraaij & Pohlmann, 1994] Kraaij, W. & Pohlmann, R. Porter's stemming algorithm for Dutch. In Noordman, L. and de Vroomen, W., editors, Informatiewetenschap, Tilburg, STINFON, 1994.
[Lancaster, 1995] Lancaster, F. W. El Control del Vocabulario en la Recuperación de Información. Ed. Universidad de Valencia, España, 1995.
[Losee, 1998] Losee, R., Text Retrieval and Filtering: Analytic Models or Performance, Kluwer, Boston, 1998.

[Martínez Méndez, 2004] Martínez Méndez, Francisco Javier. Recuperación de información: modelos, sistemas y evaluación. Murcia: KIOSKO JMC, 2004.
[Miller, 1995] Miller, G. A lexical database for English. Communication of the ACM. Vol. 38, Issue 11, pp: 39-41, Nov. 1995.
[Page et al., 1998] Page L., Brin S.. The PageRank Citation Ranking: Bringing Order to The Web, Standford Digital Library Technologies, Working Paper 1999-0120, Stanford Univ., Palo Alto, Calif., 1998.
[Ponce et al., 2008] A. Ponce, C. Deco, C. Bender. Proposal of an ontology based web search engine. En Proceedings del Workshop de Bases de Datos en el marco del XIV Congreso Argentino de Ciencias de la Computación, CACIC 2008. Chilecito, Argentina, octubre 2008.
[Porter, 1980] Porter, M. An Algorithm for Suffix Stripping. Program, 14:130–137, 1980.
[Salton, 1983] Salton, G. Introduction to Modern Information Retrieval. New York: McGraw-Hill, 1983.
[Savoy, 1999] Savoy, J. A Stemming Procedure and Stopword List for General French Corpora. Journal of the American Society for Information Science, 50:944–952, 1999.
[Swanson, 1988]. Swanson, Don R. 1988. Historical note: Information retrieval and the future of an illusion. JASIS 39(2):92–98.
[Schamber, 1996]. Schamber, Linda, Michael Eisenberg, and Michael S. Nilan. 1990. A re-examination of relevance: toward a dynamic, situational definition. IP&M 26(6):755–776.
[Schinke et al., 1996] Schinke, R., Robertson, A., Willet, P., Greengrass, M. A stemming algorithm for Latin text databases. Journal of Documentation, 52:172–187, 1996.
[Studer, 1998] Studer S, Benjamins R., Fensel D. Knowledge Engineering: Principles and Methods. Data and Knowledge Engineering, vol. 25, pp. 161-197, 1998.
[van Rijsbergen, 1979]. van Rijsbergen, C. J. Information Retrieval. Butterworths, 1979.
[Vossen, 1998] Vossen, P. Introduction to EuroWordNet. Computers and the Humanities, Special Issue on EuroWordNet, 1998.

3 — Bases de Datos y Web

Elaborado por: Cristina Bender - Claudia Deco
Universidad Nacional de Rosario
Universidad Católica Argentina, Campus Rosario

3.1 Introducción

La Web nos provee de una infraestructura global y un conjunto de estándares que soportan el intercambio de documentos. Se tiene un formato de presentación para hipertextos (*HTML*) e interfaces bien diseñadas para la recuperación de documentos, mediante el uso de técnicas de recuperación de información. Actualmente la Web es la base de datos más grande. Por otro lado, las bases de datos, nos ofrecen: técnicas de almacenamiento y lenguajes de consulta, que proveen acceso eficiente a grandes cuerpos de datos muy estructurados; modelos de datos y métodos para estructurar esos datos; y mecanismos para mantener la integridad y consistencia de los datos. Surge entonces la necesidad de un puente para poder consultar a la Web como a una base de datos. La solución es el formato XML para intercambiar datos con estructura, y un modelo de datos semiestructurados, que relaja la sintaxis de sistemas de base de datos muy estructurados. En este capítulo se describen el modelo de datos semiestructurados, el formato XML y formatos posteriores basados en XML. En el último apartado de este capítulo se lista la bibliografía utilizada para el desarrollo del capítulo. En el Capítulo Bases de Datos NoSQL, se encuentran temas adicionales a los tratados aquí.

3.2 Datos Semiestructurados

Los datos semiestructurados son datos irregulares o incompletos. Su estructura puede cambiar de forma impredecible, y los datos nuevos pueden no respetar la estructura ya existente. Son datos sin esquema o auto-descriptibles, donde la información sobre la estructura está junto con los datos. Por esto, su representación es mediante una lista de etiquetas-valor. Un ejemplo elemental es el siguiente:

{ name: {first: "Pablo", last: “Pérez"}, tel : 4494403, email: pablo@hotmail.com}

que contiene información sobre personas, sus números de teléfono y sus direcciones de e-mail. En la representación los valores pueden ser atómicos (tal el caso del teléfono) o estructuras anidadas (como ocurre con el nombre que está a su vez compuesto por el primer nombre y el apellido). Tampoco existe restricción sobre la repetición de campos (podría tenerse múltiples ocurrencias del campo teléfono si la persona tiene varios). Además, si no se tiene un dato, dicho par atributo-valor estará ausente en la representación, como se muestra en el siguiente ejemplo, donde se desconocen el apellido y el email de Marcelo:

{ person: {name: {first: "Pablo", last: “Pérez", tel: 4494403, tel: 4494423, email: pablo@hotmail.com},
person:{name: "Marcelo", tel: 4252307},}

3.3 XML (*eXtensive Markup Language*)

HTML es un lenguaje simple basado en *tags* para especificar el formato de los documentos. Cada etiqueta indica el comienzo y el final de los elementos que componen el documento. Los *tags* describen el diagrama en el que se presenta el documento, las páginas a las que enlaza el documento, y los dibujos o formas que se incluyen en él. Un ejemplo de un documento en HTML es el siguiente:

```
<HTML>
      <head>
          <title>Titulo</title>
      </head>
      <body>
          <img src="/carpeta/imagen.jpg"/>
      </body>
</HTML>
```

XML sirve para describir, estructurar y almacenar información. Esto es importante para el intercambio de una amplia variedad de datos en la Web. No provee instrucciones de cómo se presentan los datos, que es lo que hace HTML. Cada usuario puede crear su propio lenguaje para el formato de datos siguiendo determinadas reglas. XML no impone restricciones semánticas, y las estructuras pueden ser anidadas a cualquier profundidad. Esta estructuración ayuda a ordenar contenidos. Algunos ejemplos de XML son:
Ejemplo 1:

<datos>
<nombre>José</nombre>
<apellido>Pérez</apellido>
<edad>24</edad>
</datos>

Ejemplo 2:

<datos nombre=“José” apellido=“Pérez” edad=“24”/>

Ejemplo 3:

<datos edad=“24”>
<nombre>José</nombre>
<apellido>Perez</apellido>
</datos>

En los tres ejemplos anteriores se representa la misma información. El nombre de la persona, José Pérez, y su edad, 24 años.

XML se apoya en la teoría de los datos semiestructurados. Por esto, los componentes básicos en XML son **elementos** (el texto que es el valor del campo), y **etiquetas** (que son los nombres

de los campos). Las etiquetas son definidas por el usuario, no hay etiquetas predefinidas; y se permite la repetición y la ausencia de datos.

Los **elementos** se dividen en dos tipos: vacíos y no vacíos. Toda etiqueta no vacía debe tener la etiqueta de cerrado, y todos los elementos deben anidarse correctamente. Los elementos vacíos se pueden representar como <dato></dato> o como <dato/>.

Los **atributos** representan propiedades. Se definen como pares (nombre,valor) en un *tag*. Pueden definirse múltiples atributos en un mismo *tag*, pero un mismo atributo puede definirse una única vez en un *tag*.

A un archivo XML se le puede asociar otro archivo denominado DTD (*Document Type Definition*), en el cual se describen los elementos disponibles en el documento XML. El diseño y la construcción del DTD no son triviales. Un XML puede ser utilizado con o sin DTDs. Esto introduce el concepto de Documento-bien-formado y de Documento-válido. En un documento Bien-formado los *tags* se abren y se cierran, esto es se cumplen con las restricciones sintácticas del lenguaje y se cumple con una estructura jerárquica estricta. En los documentos Válidos se tienen DTDs asociados; esto es, son documentos bien formados y que siguen una estructura y semántica determinada por un DTD. Cuando se procesa información formateada mediante XML se comprueba primero si está bien formada y, en el caso que incluya o referencie a un DTD, comprueba si se siguen sus reglas gramaticales.

Un ejemplo de un XML bien formado es el siguiente:

```
<?xml version="1.0" encoding="UTF-8" stanalong="yes"?>
<datos>
    <nombre>José</nombre>
    <apellido>Pérez</apellido>
    <edad>24</edad>
</datos>
```

Para referenciar un DTD en un documento XML se puede incluir una referencia al documento DTD en forma de URL, de la forma siguiente: <!DOCTYPE mi_dtd SYSTEM “http://.....”>

Un ejemplo de un DTD es el siguiente:

```
<!DOCTYPE db [ <!ELEMENT db (person*)>
    <!ELEMENT person (name,age,email)>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT age (#PCDATA)>
    <!ELEMENT email (#PCDATA)>]
>
```

Este DTD puede incluirse en el propio documento XML, por ejemplo de la forma siguiente:

```
<? xml version="1.0"?>
<!DOCTYPE datos[
    <!ELEMENT datos (nombre+, apellido+, sexo, direccion*, foto?)>
    <!ELEMENT nombre(#PCDATA)>
    <!ELEMENT apellido (#PCDATA)>
    <!ATTLIST nombre sexo(masc|fem)>
    <!ELEMENT dirección(#PCDATA)>
```

```
    <!ELEMENT foto EMPTY>]
>
```

Si se observa el ejemplo anterior, se ve que se tienen caracteres especiales en la definición de los datos. Estos caracteres son: + indica el uso obligatorio de este atributo y la posibilidad que sea de múltiples valores; * indica un uso opcional y múltiple; ? indica uso opcional y singular (una sola ocurrencia); y | indica uso obligatorio y singular.

Los DTD no satisfacen todas las necesidades de XML. Para solucionar esto ha surgido **XMLSchemas** que es más riguroso y sofisticado para tratar la estructura y semántica dentro de un documento XML. Esto es, son una ampliación y mejora a los DTDs. Indican tipos de datos, números máximos y mínimos de ocurrencias y otras características específicas. Usan sintaxis XML.

Sea el siguiente ejemplo:

```
<Schema xmlns="urn:Schemas-Microsoft-com:xml-data"
Xmlns:dt="urn:schemas-Microsoft-com:Datatype">
<Attribute Type name='id dt:type='string' required='yes' />
<Element Type name='nombre' content='textOnly' />
<Element Type name='persona' content='mixed' > <attribute type='id'/> <element type='nombre'/>
</Element Type>
<Element Type name='nombre' content='eltOnly' > <element type='persona'/> </Element
Type>
</Schema>
```

En el ejemplo anterior, xml-data indica que es un esquema y no otro documento XML cualquiera; Datatypes permite definir tipos a elementos y atributos, usando el prefijo "dt"; Element Type define el tipo y contenido de un elemento, también sub-elementos; AttributeType asigna tipo y condiciones a los atributos; attribute declara qué atributos definidos en AtributeType son atributos de un elemento determinado; y element declara qué elementos declarados en Element Type puede aparecer como contenido de otro.

Existen lenguajes de consulta a archivos XML, por ejemplo XQuery. Este lenguaje de consulta se basa en operadores de búsquedas para realizar consultas en diferentes tipos de documentos. Utiliza notación XML para definir consultas y manipular los resultados. Cada entrada y salida a una consulta es una instancia de un modelo de datos.

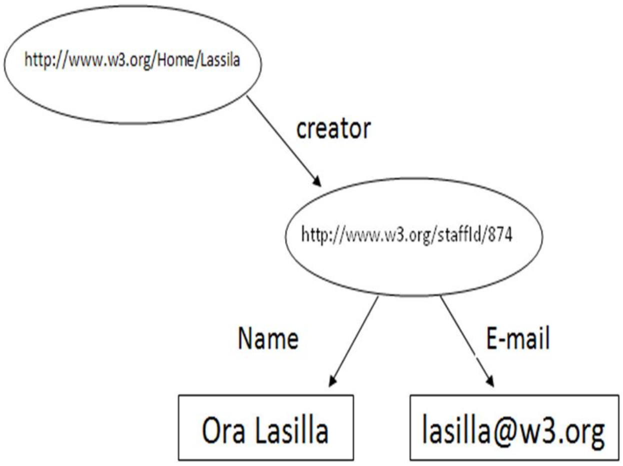
3.4 RDF (*Resource Description Framework*)

El objetivo de RDF es complementar a XML especificando la semántica para las BD XML de una forma normalizada e interoperable, permitiendo intercambiar y reutilizar información en la Web. RDF asocia información sobre el contenido de los recursos Web. Sus características principales son: Independencia, cualquier diseñador puede inventar propiedades; Intercambio, dado que las sentencias se escriben en XML y pueden usarse para intercambiar información; y escalabilidad, porque las sentencias son fáciles de manejar y usar para buscar objetos aún en volúmenes muy grandes. Es recomendado por el consorcio W3C como estándar para los metadatos.

RDF consiste en tres tipos de Objetos: Recursos, es decir, cualquier objeto web identificable como URL; Propiedades, o sea los aspectos específicos, características, atributos o relaciones utilizadas para describir recursos; y Sentencias del tipo sujeto – predicado - objeto, que son el conjunto de un recurso, un nombre de propiedad y el valor de esa propiedad. Un ejemplo

de sentencia es “Ora Lassilla es el *creator* del recurso http://www.w3.org/Home/Lassila” , donde el sujeto es http://www.w3.org/Home/Lassila, el predicado es “*creator*”, y el objeto es “Ora Lassilla”.

Esto puede representarse en un diagrama de nodos y arcos. Para observar esto, supongamos que existe http://www.w3.org/staffId/847 y que se quiere representar la sentencia “El individuo al que se refiere el identificador de empleado 874 se llama Ora Lassila y tiene la dirección de correo lassila@w3.org. Ese individuo creó el recurso http://www.w3.org/Home/Lassila”. El diagrama de nodos y arcos es entonces:



Y la sintaxis correspondiente es:

```
 <rdf:RDF>
   <rdf:Description about="http://www.w3.org/Home/Lassila">
       <s:Creatorrdf:resource="http://www.w3.org/staffId/85740"/>
   </rdf:Description>
   <rdf:Description about="http://www.w3.org/staffId/85740">
       <v:Name>OraLassila</v:Name>
       <v:Email>lassila@w3.org</v:Email>
   </rdf:Description>
</rdf:RDF>
```

3.5 Ontologías y OWL (*Ontology Web Language*)

Una ontología es una especificación explícita de una conceptualización [Gruber, 1993]. Según las recomendaciones del consorcio W3C, una ontología describe los términos usados para describir y representar un área de conocimiento. Las ontologías son usadas por personas, bases de datos, y aplicaciones que necesitan compartir información de dominio. Incluyen definiciones de conceptos básicos de dominio, utilizables por computadoras, y las relaciones entre ellos. Codifican el conocimiento en un dominio y también conocimiento que abarca varios dominios, con el objetivo de hacer este conocimiento reusable. OWL es la solución que propone el W3C para describir ontologías.

Los elementos constitutivos de una ontología son Individuos, Clases, Atributos, Relaciones y Eventos. Los Individuos son los objetos básicos.

Los Eventos manifiestan el cambio de atributos o relaciones en los objetos.

Las Clases son conjuntos, colecciones o tipos de objetos. Pueden contener individuos, otras clases, o una combinación de ambas. Una clase puede pertenecer a sí misma y puede existir una clase que contenga a todas las clases.

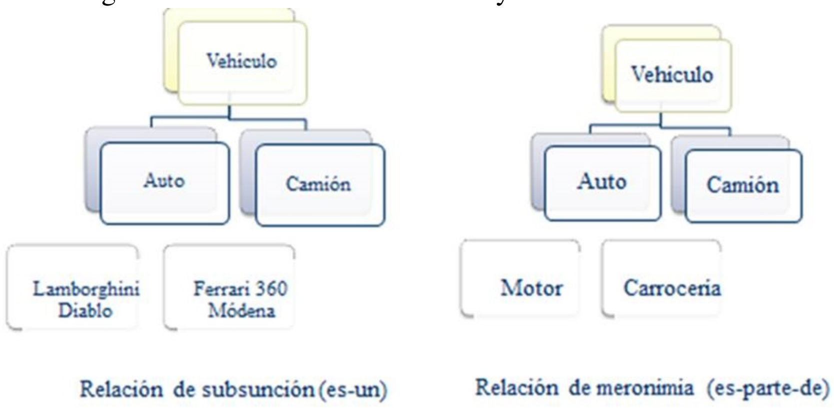
Los Atributos son propiedades, características o parámetros que los objetos pueden tener y compartir. Cada atributo tiene un nombre y un valor y se usa para guardar información específica para dicho objeto. Por ejemplo, sea el objeto Lamborghini Diablo correspondiente a un auto. Sus atributos pueden ser:

Nombre: Lamborghini Diablo
Número-de-puertas: 2
Motor: {V12 5.7 Litros, V12 6.0 Litros}
Diseñador: Marcello Gandini

Las Relaciones son las formas en las que los objetos pueden ser vinculados uno a otro. Una relación es un atributo de un objeto cuyo valor es uno o más objetos. El poder de las ontologías proviene de la habilidad para expresar estas relaciones. Son estas relaciones las que definen la semántica del dominio correspondiente. Por ejemplo, para el objeto Lamborghini Diablo, puede existir un atributo Sucesor cuyo valor es Lamborghini Murciélago.

Existen distintos tipos de relaciones. Un primer tipo es la Relación de subsunción. Subsunción proviene del verbo subsumir que se define como "Considerar algo **como parte** de un **conjunto más amplio** o como caso particular sometido a un principio o norma general". Esta relación expresada por la frase "es un", permite la jerarquización de las clases, según el nivel de abstracción de cada una. Por ejemplo, el objeto Lamborghini Diablo **es un** auto y un auto es un vehículo. Si hay relación de subsunción, estamos hablando de una taxonomía. Otro tipo de Relación, es una relación de meronimia (es-parte-de). Este tipo de relación permite la partición de un objeto en sus partes constitutivas. Por ejemplo, el motor **es parte de** un auto. En la figura 3.1 se esquematizan estos dos tipos de relaciones.

Figura 3.1: Relación de subsunción y Relación de meronimia



Actividad para el alumno: Analizar si es posible describir una ontología utilizando RDF. Indicar y analizar las posibles limitaciones.

El lenguaje estándar para la descripción de ontologías en la Web es OWL. Este lenguaje ha sido desarrollado por el Web Ontology Group del consorcio W3C. En OWL, los conceptos se llaman clases, mientras que los roles se denominan propiedades. En OWL, toda clase depende

de la clase “madre” owl:thing (es decir, “todo es una cosa”). Usa XML como lenguaje de base.

Un ejemplo de escritura en OWL, donde se indica que la clase “Female” es una subclase de “Animal” y que es una clase cuyos elementos son disjuntos de aquellos de la clase “Male”, es el siguiente:

```
<owl: Class rdf:ID="Female">
     <rdfs:subClassOF rdf:resource:"#Animal"/>
     <owl:disjointWith rdf:resource:"#Male"/>
<owl: Class>
```

Un ejemplo más completo de la sintaxis de OWL, donde se especifican condiciones necesarias y suficientes para determinar que una cosa es *Auto* es el siguiente:

```
<owl:equivalentClass>
    <owl:Class>
    <owl:intersectionOf rdf:parseType=Çollection">
    <owl:Restriction>
    <owl:onProperty>
    <owl:FunctionalProperty rdf:about="#TieneMotor/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="www.w3.org/XMLSchema#boolean”>true</owl:hasValue>
    </owl:Restriction>
    <owl:Restriction>
    <owl:onProperty>
    <owl:FunctionalProperty rdf:about="#Cantidad_De_Ruedas/>
    </owl:onProperty>
    <owl:hasValue rdf:datatype="www.w3.org/XMLSchema#int”>4</owl:hasValue>
    </owl:Restriction>
    </owl:intersectionOf>
    </owl:Class>
</owl:equivalentClass>
```

Un editor de ontologías posibilita abstraerse de la sintaxis extensa y compleja de RDF/XML. Un editor diseñado en la Universidad de Stanford es Protègé.

Actividad para el alumno: Realizar un relevamiento de editores de ontologías y evaluar las características distintivas de cada uno.

3.6 La Web Semántica

El objetivo de la Web Semántica es permitir el procesamiento automático de la información que se encuentra disponible. La clave del problema es que la web actual representa la información utilizando lenguaje natural con muy poca estructura (se utiliza el lenguaje de marcado HTML, gráficos, videos, etc.). El lenguaje natural es comprendido por humanos pero complejo para ser procesado automáticamente.

Las alternativas son dos. O bien tener máquinas más inteligentes, esto es enseñar a las computadoras a comprender el significado de la información que hay en la web, utilizando técnicas y herramientas tales como procesamiento de lenguaje natural, reconocimiento de imágenes, etc. O bien tener Información más inteligente, lo cual significa representar la información de modo que sea sencilla de comprender a las máquinas, expresando los contenidos en un formato procesable automáticamente, por ejemplo mediante metainformación.

La web semántica consiste de dos partes: Formatos comunes para la integración y combinación de datos generados por diversas fuentes, y Lenguajes para registrar la forma en que los datos se relacionan con objetos del mundo real. La idea es permitir que los datos sean procesados por máquinas (agentes) independientemente de la aplicación que los genera, así como poder navegar a través de bases de datos que traten "sobre el mismo tema". Para lograr una web más semántica, Tim Berners-Lee promueve el desarrollo de la web con conocimientos, y existen organizaciones que se encargan de estandarizar lenguajes y herramientas para dar semántica a la web, tales como SemanticWeb y W3C.

Los buscadores (Google, Yahoo!, etc.) realizan la búsqueda mediante palabras clave en el HTML de las páginas. Con la tendencia de implementar el uso de metadatos dentro del código HTML, agentes de búsqueda pueden encontrar la información de forma precisa y realizar inferencias automáticamente buscando información relacionada, utilizando ontologías.

Por esto, la importancia de las ontologías en la web se aprecia con la aparición de agentes de búsqueda de información, que explotarán el conocimiento en las páginas web, serán capaces de interpretar los esquemas ontológicos y axiomas de diferentes dominios, mantendrán la consistencia de las instancias que se inserten en las páginas web siguiendo los esquemas ontológicos definidos, y realizarán búsqueda con inferencias utilizando los axiomas. Para hacer que los datos sean procesables por máquinas, se necesitan nombres no ambiguos para los recursos (que pueden también ligar datos con entidades del mundo real), esto es URIs; un modelo de datos comunes para acceder a, conectar y describir los recursos, tal como RDF; tener acceso a estos datos con un lenguaje de consulta, por ejemplo SPARQL; posibilidad de definir vocabularios comunes (compartidos), con por ejemplo un lenguaje como OWL; lógicas para razonamiento, por ejemplo con reglas implementadas en OWL.

3.7 Bibliografía

[Abiteboul et al., 1999] Abiteboul S., Buneman P., Suciu D. Data on the web. Morgan Kaufmann,1999

[OWL] OWL web ontology language guide. W3C Recommendation, 2004. Disponible en www.w3.org/TR/2004/REC-owl-guide-20040210/

[Baeza, 1999] Baeza-Yates R., Ribeiro-Neto B. Modern Information Retrieval. Addison-Wesley, 1999

[RDF] RDF Semantics. Disponible en www.w3.org/TR/rdf-mt/

[Protege] The Protégé Editor and Knowledge Acquisition System. Universidad de Stanford. Disponible en protege.stanford.edu/.

[SWeb] The Semantic Web: An Introduction. Disponible en infomesh.net/2001/swintro/#whatIsSw

[Gruber, 1993] Gruber T. R. A Translation Approach to Portable Ontology Specifications. Disponible en tomgruber.org/writing/ontolingua-kaj-1993.pdf

[W3C] W3C Semantic Web Activity. Disponible en www.w3.org/2001/sw/

4 — Sistemas de ayuda a la toma de decisión

Elaborado por: Cristina Bender y Claudia Deco
Universidad Nacional de Rosario
Universidad Católica Argentina, Campus Rosario

4.1 Introducción

Información es todo lo que reduce la incertidumbre sobre algún aspecto de la realidad y, por lo tanto, permite tomar mejores decisiones. Los requerimientos de información para la toma de decisión han evolucionado a lo largo del tiempo. En un primer momento, las consultas se orientaban a saber qué pasó, lo cual se podía responder a partir de la generación de reportes y con el uso de consultas predeterminadas y eventualmente consultas ad-hoc, directamente sobre el sistema de bases de datos transaccional. Luego, la inquietud principal fue saber por qué pasó, esto es analizar los hechos ocurridos, lo cual llevó a aumentar la posibilidad de realizar consultas ad-hoc y también a incluir modelos analíticos. En la actualidad la pregunta que predomina es qué pasará, lo cual incrementa la necesidad de utilizar tanto consultas ad-hoc como modelos analíticos que, utilizando datos de muchos hechos pasados, permitan analizar la evolución del aspecto de interés de la realidad. Algunas aplicaciones donde la información obtenida ayuda a la toma de decisiones son el marketing, por ejemplo para realizar una segmentación del mercado; el análisis de riesgo financiero o de riesgo crediticio; la logística de la empresa, por ejemplo para optimizar los niveles de producción; el control de vectores en el área salud; entre otros.

Generalmente, la información que se quiere obtener sobre un cierto dominio de la organización se encuentra en bases de datos y otras fuentes muy diversas, tanto internas a la organización como externas. Muchas de estas fuentes internas son las que se utilizan para el trabajo diario, esto es bases de datos transaccionales que permiten realizar el procesamiento en línea de las transacciones (OLTP, *On Line Transaction Processing*). Un sistema de procesamiento de transacciones posee gran nivel de detalle y está diseñado para soportar actualizaciones consistentes y sin redundancia, a partir de un conjunto de tablas normalizadas. Pero, la información que necesita el directivo de una organización para tomar decisiones, no puede obtenerse fácilmente de estas bases de datos por distintos motivos, tales como la dificultad para navegar por toda la información de la organización, la necesidad de incorporar el entorno respecto a la información obtenida, los datos que contienen sólo reflejan el estado actual de las transacciones; y existe información de la organización que no se encuentra en ellas (por ejemplo, e-mails, documentos, planillas, etc.

Para la toma de decisiones se necesita tener estos datos todos juntos, y esto se logra utilizando almacenes o repositorios de datos (DW, *Data Warehouse*). Este es el tema principal de este capítulo. De estos datos almacenados, es de interés extraer conocimiento. El conocimiento evidente es fácilmente recuperable mediante consultas SQL, si los datos están almacenados en una base de datos relacional. Pero existe dificultad para expresar en SQL algunas consultas

o realizar análisis estadísticos, lo cual plantea la necesidad de extensiones al SQL, el uso de software estadístico, o realizar un tratamiento multidimensional de los datos para lo cual se utilizan herramientas OLAP (de la sigla en inglés *On Line Analytical Processing*, tratamiento analítico en línea). En este capítulo se presenta una breve introducción a este tipo de análisis. Finalmente, existe conocimiento oculto, que brinda información muy valiosa y desconocida. Este tipo de conocimiento es recuperable mediante la aplicación de técnicas de minería de datos (*Data Mining*). Este tema se trata en el capítulo Minería de Datos.

4.2 Almacén de Datos (*Data Warehouse*)

Según (Silberschatz et al., 2006), un almacén de datos es un depósito de la información a partir de varias fuentes, guardada según un esquema unificado en un único lugar.

Según (Inmon, 2002) es una colección de datos orientada a un tema, integrada, no volátil, y variante en el tiempo. Orientada a un tema se refiere a que se organizan y presentan los datos desde la perspectiva del usuario, por ejemplo las ventas (cantidad de ventas o montos de ventas) pero no interesa cada una de las transacciones. La colección es integrada significa que se incorporan al repositorio datos obtenidos de distintas fuentes a lo largo de muchos años y por diversas aplicaciones. Decir que la colección es no volátil significa que los datos no se borran o modifican (como ocurre habitualmente en un sistema transaccional), sino que se mantiene el dato original y se inserta el dato nuevo. Esto está ligado a la idea de que la colección es variante en el tiempo. Además, maneja un gran volumen de datos puesto que generalmente contiene todos los datos históricos de la organización. Estos datos se resumen y agregan para que sean más comprensibles para el usuario.

Entonces, con un adecuado repositorio de datos se puede recolectar y organizar la información analítica necesaria y disponer inmediatamente de ella en diversos formatos, tales como tablas, gráficos, reportes, etc.; además se puede analizar los datos desde diferentes perspectivas, llamadas dimensiones del negocio. Un sistema para la toma de decisiones debe soportar análisis complejos en estos grandes volúmenes de datos. Así, se requieren distintas técnicas de diseño a la de un sistema transaccional, por ejemplo, desnormalizar los datos, y distintos mecanismos para el procesamiento de las consultas orientados a consultas de agregación.

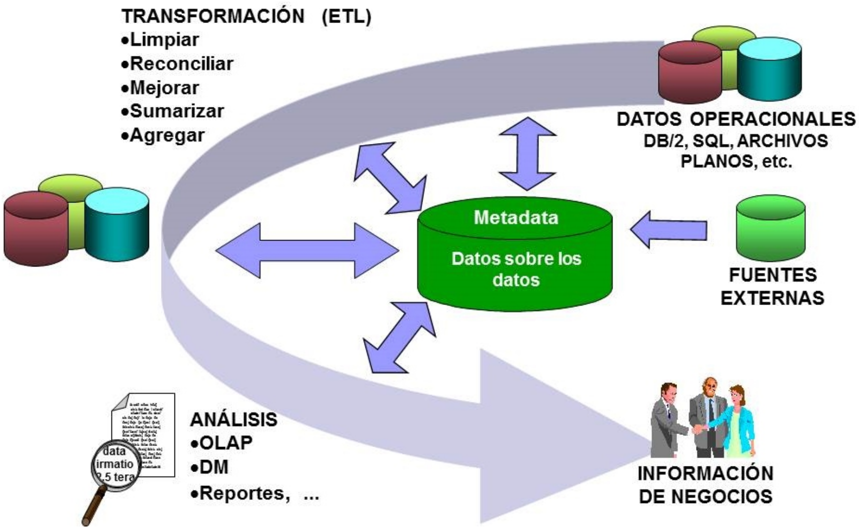
Algunas características de un almacén de datos son: es utilizado por profesionales de la dirección de la organización, la actividad más importante que se realiza sobre los datos es el análisis orientado a la decisión estratégica por lo que predomina la consulta sobre la actualización o inserción de datos; interesan los datos históricos y los datos agregados; se tiene una visión multidimensional de los datos; las bases de datos son grandes (del orden de TeraBytes); y las consultas no son previsibles. El desafío para una implementación exitosa de un almacén de datos es no olvidar que nadie sabe qué tipo de preguntas harán los usuarios, debe responder cualquier pregunta, devolver cualquier dato, y hacerlo en cualquier momento. Por esto, para garantizar el éxito no hay que restringir las preguntas de los usuarios, los datos a los que los usuarios tienen acceso, ni la cantidad de datos a los que los usuarios tienen acceso.

Dadas estas características y teniendo en cuenta que algunos de los análisis utilizan muchos recursos y tiempo (son consultas “pesadas”), generalmente se independiza del sistema transaccional para no afectar la carga de trabajo del sistema transaccional.

Existen tres clases de almacenes de datos, según la información que contengan. Si el almacén representa la información de toda la organización, se tiene un almacén empresarial (*Enterprise Warehouse*). En el caso de contener un subconjunto de la información de la organización útil para grupos específicos de usuarios (por ejemplo, para el sector de ventas de una empresa), se denomina *Data Mart*. En algunos casos puntuales, se genera un conjunto de vistas sobre los datos transaccionales y se denomina *Virtual Warehouse*.

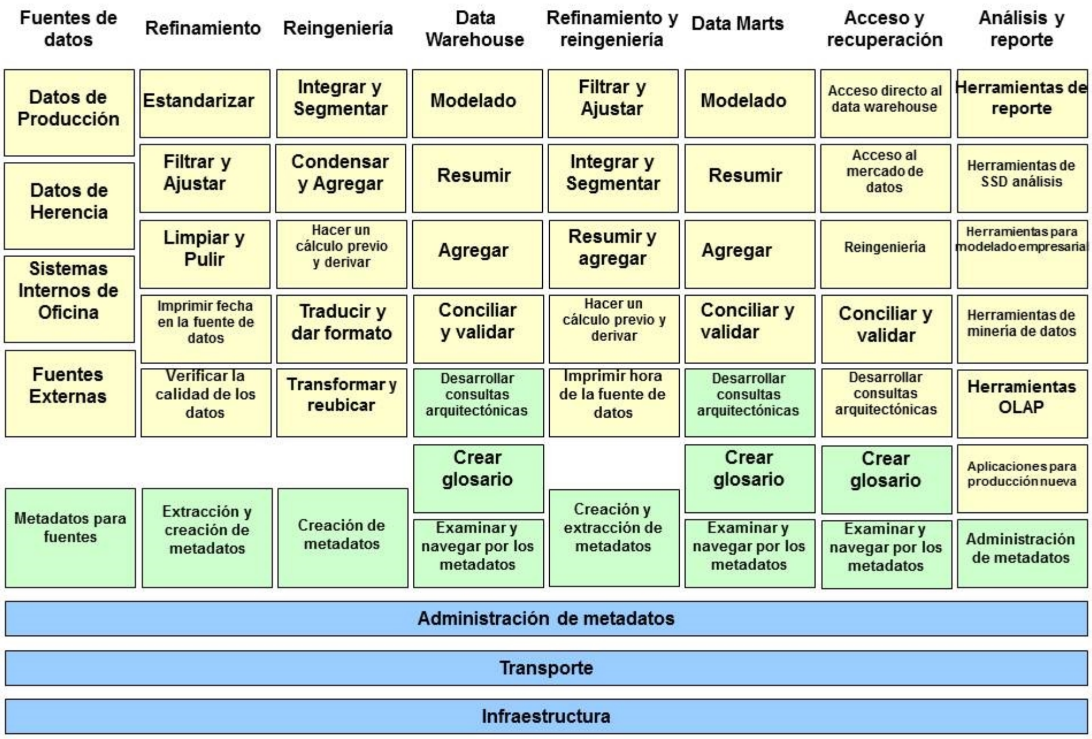
4.2.1 Preparación del almacén de datos

La preparación del almacén de datos se denomina Data Warehousing y es un proceso, no un producto. Es una técnica que permite ensamblar y administrar adecuadamente datos provenientes de distintas fuentes, con el propósito de contestar preguntas de negocios y soportar la toma de decisiones. Así, el proceso de Data Warehousing puede graficarse de la forma siguiente:



donde, a partir de las distintas fuentes de datos, se procede a un proceso de preparación de los datos (Transformación ETL), lo cual permite tener los datos disponibles adecuadamente para efectuar el análisis (por ejemplo, con herramientas OLAP, de minería de datos, de reporte, etc.) y obtener así la información de negocio necesaria. Todo esto apoyado por un conjunto de metadatos, es decir datos sobre los datos.

Una arquitectura en capas del almacén de datos, que permite visualizar los distintos procesos necesarios es la de la figura siguiente:



4.2.2 Preparación de los datos

La preparación de los datos para ser volcados a un almacén de datos consta de varias etapas. La primera es la migración de los datos, esto es mover los datos desde los sistemas operacionales de la organización a un área de trabajo (*data staging area*), donde los datos se limpian y se transforman antes de ser cargados al almacén. En forma preventiva hay que evitar mover datos innecesarios para la toma de decisiones. Una segunda etapa consiste en la limpieza de los datos (*data cleaning*), esto es corregir, estandarizar y completar los datos, identificar datos redundantes, identificar valores atípicos (*outliers*), e identificar valores perdidos (*missings*), para establecer cuáles serán las reglas de negocios a aplicar en estos casos (por ejemplo, eliminar los datos atípicos o los datos perdidos). Un dato atípico podría ser tener valores negativos para un dato referido a cantidad de lluvia caída. Un dato perdido puede ser la falta de la fecha de nacimiento de una persona. Las políticas de cómo proceder en estos casos son preparadas por el usuario conocedor del negocio.

Luego de estas etapas, se procede a efectuar los procedimientos ETL de extracción, transformación y carga (*loading*) de los datos al repositorio. La extracción refiere a capturar y copiar los datos requeridos de uno o más sistemas transaccionales o fuentes de datos, colocándolos en un archivo intermedio con un formato definido. El proceso de transformación corresponde a leer estos archivos intermedios, construir registros en el formato del almacén de datos y crear un archivo de salida conteniendo todos los registros nuevos a ser cargados en el almacén, realizando las transformaciones necesarias (por ejemplo, fusionar campos o datos homónimos, o cambiar la representación de los datos). Finalizados estos dos procedimientos se procede a la carga del repositorio.

La última etapa, antes de proceder a colocar el almacén de datos en producción, corresponde a la conciliación y validación. La validación de la carga permite identificar problemas en los datos que no se detectaron en las etapas anteriores. Esta validación se puede realizar en forma completa al final del proceso, o por etapas a medida que se cargan los datos. Los controles incluyen reportes que comparan los datos del almacén con las fuentes de datos operacionales a través de, por ejemplo, totales de control, cantidad de registros cargados, valores originales versus valores transformados, etc.

Un problema que se presenta en el proceso de *datawarehousing* es el momento y manera de recoger los datos, esto ha originado la propuesta de dos arquitecturas: una orientada a los orígenes de datos, donde las fuentes de datos transmiten la información nueva al almacén; y una orientada a los destinos de datos, donde el almacén de datos solicita los datos a las fuentes. Otros problemas son, por ejemplo, definir el esquema que debe utilizarse, establecer qué datos se deben resumir, cómo se propagan las actualizaciones de los esquemas de los orígenes de datos, entre otros.

Actividad para el alumno: Realizar unrelevamiento de problemas vinculados con la creación del repositorio y las propuestas para solucionarlos. Realizar una crítica a estas propuestas.

4.3 Modelo de datos

El modelo relacional utilizado en los sistemas transaccionales tiene como objetivos garantizar la integridad de los datos y eliminar cualquier tipo de redundancia en los datos. Pero tiene problemas: para el usuario la legibilidad es limitada, puede haber dificultad para las herramientas de consulta en el acceso a los datos y frustra el principal atractivo del repositorio de datos que es realizar una recuperación de información intuitiva y con alto rendimiento. Estos problemas indican que es necesario otro modelo: el Modelo Multidimensional (MMD), el cual brinda una vista multidimensional del repositorio, afectada por el diseño de la base de datos, las herramientas

front-end, y los motores OLAP.

En este modelo, se tiene un conjunto de medidas numéricas que son los objetos de análisis, por ejemplo ventas, duración de llamadas, etc. Asociadas a las medidas, se tienen las dimensiones de análisis, que proveen el contexto a las medidas, y se describen mediante atributos, por ejemplo, (tipos de) clientes, llamadas telefónicas, etc.

El modelo multidimensional es una técnica de diseño lógico que busca presentar la información en un marco estándar e intuitivo que permite un acceso de alto rendimiento. Consiste de *Áreas temas,* que corresponden al ámbito de implementación del almacén; *Dimensiones*, esto es una colección de miembros o entidades del mismo tipo (por ejemplo, para una organización vinculada a la telefonía, la duración de una llamada, si la llamada es local o al exterior, etc,); *Hechos* (cada mes, la organización hace llamadas locales y/o al exterior), *Medidas* que son cuantificadores del desempeño de un ítem (qué cantidad de llamadas al exterior realizó el sector X de la organización en un mes / año dados); y *Jerarquías*, esto es un conjunto de miembros de una misma dimensión (por ejemplo, para el tiempo, se puede tener una jerarquía: año - trimestre - mes).

Entonces, el Modelo Multidimensional se centra en dos conceptos. Las dimensiones que son categorías sintácticas que permiten analizar la información desde distintos enfoques, donde cada dimensión está organizada en una jerarquía de niveles, correspondientes a dominios de datos de diferente granularidad. Las tablas de hechos (*fact tables*) que son funciones que toman coordenadas y devuelven medidas que nos brindan la información.

El tipo de respuesta a las consultas suele tener la forma de un cubo de datos, como se muestra en el ejemplo siguiente. Para la siguiente especificación:

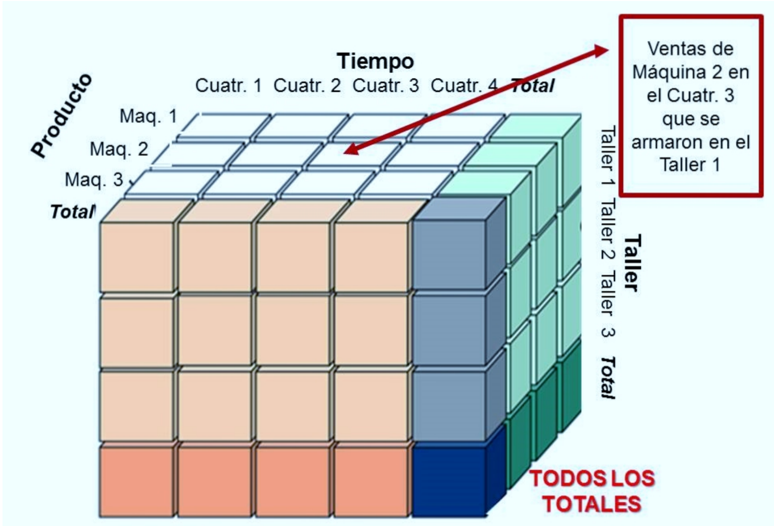
Dimensiones:	(Tiempo, Producto, Taller)
Jerarquías:	(Año -> Semestre -> Mes -> Semana), (Categoría -> Línea)
Elementos:	(2006, 2007, ..., S1-06, ..., Ene-06, ..., 200625....), (Todos, Máquinas, Refacciones, Máquinas caras, Máquinas baratas, Máquina 1,...) (Todos, Taller 1, Taller 2, ...)
Hechos:	(Ventas, Inventario, Defectos, Devoluciones)

La tabla para las dimensiones Tiempo y Producto podría tener la forma siguiente:

Tiempo	Productos	Ventas	Inventario	Defectos	Devoluciones
2006	Todos	1000	200	50	10
Ene-06	Maquina 1	10	100	10	10

donde todos los valores deben estar precalculados.

Si se consideran las tres dimensiones (Tiempo, Producto y Taller), las ventas se pueden representar con un cubo de datos (*Data Cube*):



La técnica de diseño más popular para este modelo es el Esquema *Estrella*, donde un único objeto en el centro (tabla de hechos), está conectado radialmente con otros objetos (tablas de dimensiones) formando una estrella (Kimball, 1996). En este esquema la tabla de hechos contiene claves de las dimensiones y atributos. En las puntas de la estrella se encuentran las tablas de dimensión que contienen los atributos de cada dimensión y se pueden utilizar como los criterios de búsqueda para SQL. Las tablas de hechos y dimensiones están relacionadas por claves foráneas. Este esquema permite utilizar herramientas OLAP para el acceso a la información.

Una extensión al esquema estrella es el Esquema de *Constelación*. Aquí cada tabla de dimensiones puede estar asociada con más de una tabla de hechos. Una ventaja de esta extensión es que se hace un mejor uso del almacenamiento evitando la redundancia.

Otra extensión es el Esquema *Copo de Nieve*, donde cada punta de la estrella explota en más puntas. En este caso las tablas de dimensión se encuentran más normalizadas para mejorar la performance; y en cada dimensión se almacenan jerarquías de atributos.

Abelló, Samos y Saltor, en su reporte presentan una clasificación de las propuestas para el modelado multidimensional en cuatro grupos teniendo en cuenta las fases de diseño. El grupo que llaman *Nivel Conceptual*, tiene como característica la de proveer conceptos cercanos a la manera en el que el usuario percibe los datos, es decir, son propuestas que tratan de representar cómo los usuarios perciben el cubo multidimensional, sin prestar atención al formalismo. El grupo *Nivel Lógico* considera si existe una dependencia del tipo de sistema de gestión de bases de datos usado en la implementación, es decir son propuestas cuyas construcciones son orientadas a un tipo dado de sistema de gestión de bases de datos. El grupo llamado *Nivel Físico* tiene en cuenta si existe una dependencia de la especificación del sistema de gestión de bases de datos y describe como son guardados los datos, y explica como debe ser implementado el cubo. Finalmente engloban en un cuarto grupo, *Nivel Formal*, a aquellas propuestas que definen un álgebra o un cálculo multidimensional.

Los datos para trabajar se conservan en servidores OLAP, que permiten definir y navegar un modelo multidimensional. Las dos arquitecturas predominantes son ROLAP y MOLAP, aunque también existe un enfoque híbrido. En la arquitectura MOLAP (Multidimensional OLAP) se trabaja directamente mediante un motor de almacenamiento multidimensional, conformado por *arrays* propietarios. Los servidores MOLAP soportan la visión multidimensional de datos y poseen una excelente performance. Si se utilizan bases de datos relacionales como servidores de datos, el modelo y sus operaciones deben ser mapeados a relaciones y consultas SQL y se tiene la arquitectura ROLAP (Relational OLAP). En esta arquitectura se utilizan diversas técnicas de materialización de vistas, pudiendo acceder a las tablas originales cuando sea necesario. En

algunos casos, se combinan ambas arquitecturas en una arquitectura híbrida (Hibrid OLAP) logrando menores necesidades de espacio de almacenamiento que MOLAP, y respondiendo a las consultas más eficientemente que ROLAP.

Actividad para el alumno: Realizar unrelevamiento de estos tipos de propuestas, indicando si se continúan desarrollando (o aplicando) o no lo hacen; además discutir la aceptación que tienen, realizando una crítica a cada una.

4.4 Procesamiento Analítico en Línea (OLAP)

En el Procesamiento Analítico en Línea (OLAP) se crea nueva información a partir de los datos existentes. OLAP permite realizar un análisis multidimensional de los datos mediante la navegación asistida del usuario. Estas herramientas presentan al usuario una visión multidimensional de los datos en forma de cubo. Por ejemplo, se puede generar tablas de referencias cruzadas como la siguiente, donde se presenta la cantidad de ropa comprada por los clientes según el color de la ropa (claro u oscuro) y la talla (pequeña, mediana o grande):

	Pequeña	Mediana	Grande	totales
Claro	9	35	10	53
Oscuro	20	10	5	35
Total	28	45	15	88

En este ejemplo, tenemos datos bidimensionales, basados en dos atributos: color y talla. Una tabla cruzada, como esta, no puede generarse mediante una sola sentencia SQL, dado que los totales se toman en varios niveles diferentes (para cada cruce de tipo de color y tamaño de talla, según el tipo de color para todas las tallas, según el tamaño de talla independientemente del tipo de color). Esto se traduce en trabajar con un cubo de 2 (dos) dimensiones. Una relación n-dimensional, para n atributos, generará un n-cubo de 2n vértices. Por ejemplo, para la relación Ventas (color, talla, precio), con 3 atributos, tendríamos un cubo como se mostró en el punto anterior.

En los sistemas de gestión de bases de datos relacionales, existen extensiones del SQL con un operador CUBE en la cláusula GROUPBY. Así para el ejemplo de la tabla anterior, se puede escribir:

SELECT color talla, sum(numero) FROM ventas
GROUP BY color, talla **WITH CUBE**.

Para que el usuario pueda obtener información de un sistema multidimensional mediante OLAP, las herramientas deben tener ciertas funciones para el manejo de datos. Una primer función para explorar los hechos hacia niveles más detallados de la jerarquía de dimensiones (*Drill Down*), y una función que permita realizar la exploración hacia los niveles más altos de la jerarquía (*Drill Up*). Otra función que permita definir un subconjunto del hipercubo especificando qué dimensiones y medidas interesan analizar, esto es rebanar (*Slice*) el hipercubo aplicándose una única restricción a una sola dimensión, mediante la elección de un miembro en particular. Funciones (*Dice*) que permitan aplicar más de una restricción, ya sea para una misma dimensión o para varias dimensiones, que correspondería a la combinación de varias funciones Slice. Una función (*Roll Up*) que permita explorar los hechos iterativamente hacia el nivel más alto de agregación. Finalmente alguna función (*Drill Through*) que permita acceder a los datos descriptivos.

Algunas herramientas OLAP comerciales son Microsoft SQL Server OLAP Services, BIXL, Oracle 9i, Power OLAP, y Thinking Networks AG b2Brain.

Actividad para el alumno: Realizar un relevamiento actualizado de herramientas OLAP comerciales, y compararlas. Relevar herramientas OLAP open source y compararlas. Analizar en todos los casos la facilidad de uso por parte de un usuario final, la cantidad de datos que se manejan, la arquitectura (ROLAP, MOLAP, HOLAP), el uso de datos originales, la velocidad de respuesta, y otras características propuestas por el alumno.

4.5 Problemas y temas pendientes

Algunos de los temas de investigación y desarrollo en almacenes de datos están relacionados con problemas referentes al diseño y modelado, el procesamiento de las consultas, la integración de los datos, el mantenimiento del repositorio, la visualización en OLAP, la calidad de los datos, entre otros. Además, con el avance de la tecnología referente a los dispositivos móviles y con la visión de la web como un gran repositorio de información, surgen desafíos tales como la aplicación de técnicas OLAP al sitio Web de la organización, publicar el *data warehouse* en la Web, OLAP para datos XML, etc.

Actividad para el alumno: Realizar un relevamiento actualizado de estos problemas. Indicar qué grupos de investigación y/o empresas están trabajando en ellos y evaluar las propuestas de solución. ¿Han surgido otros problemas? ¿Cuáles?

4.6 Bibliografía

[Abelló et al., 2000] Abello, A., Samos, J., Saltor, F. A Data Warehouse Multidimensional Data Models Clasification. Technical Report LSI- 2000-6. Dept. Lenguages y Sistemas Informáticos, Universidad de Granada, 2000.

[Codd et al., 1993] Codd E.F., Codd S.B. and Salley C.T. Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate. Codd & Date, Inc. Technical Report. 1993.

[Golfarelli et al., 2004] Golfarelli M., Rizzi S. and Cella I.. Beyond Data Warehousing: What's Next in Business Intelligence. In: Proceedings of the 7th ACM international workshop on Data warehousing and OLAP, pp. 1-6. Washington DC, USA. 2004.

[Inmon, 2005] Inmon W.H. Building the Data Warehouse. 4ta ed. John Wiley and Sons, 2005.

[Kimball, 1996] Kimball, R. The Data Warehouse Toolkit. John Wiley & Sons, 1996. (ver también ediciones posteriores). Ralph Kimball es miembro fundador del Kimball Group, dedicado a Inteligencia de Negocios (Business Intelligence) y es el primero que trabajó con la idea del modelado dimensional, se sugiere consultar la página www.kimballgroup.com/

[Silberschatz et al., 2006] Silberschatz A., Korth H. F., Sudarshan S. Fundamentos de bases de datos. 5ta ed. McGraw-Hill / Interamericana de España, S.A., 2006.

[Simitsis et al., 2005] Simitsis A., Vassiliadis P. and Sellis T. State-Space Optimization of ETL Workflows. IEEE Transactions on Knowledge and Data Engineering. Vol. 17, Issue 10, pp. 1404–1419. October, 2005.

5 — Minería de Datos

Elaborado por: Julio Cesar Ponce Gallegos
Universidad Autónoma de Aguascalientes, México

OBJETIVO

Este capítulo tiene como objetivo establecer un punto de partida para el estudio sobre el análisis de la información en grandes volúmenes de datos a través del uso de la Minería de Datos. El conocimiento que adquiera el estudiante podrá ser autoevaluado mediante la aplicación de un cuestionario al final del capítulo y el uso de ejercicios.

RESUMEN DEL CAPÍTULO

A lo largo de este capítulo, se hará una descripción de de los orígenes e historia del descubrimiento del conocimiento y la minería de datos, así como de algunas contribuciones que esta área ha heredado de otras disciplinas. También se discutirán algunas de las técnicas más comúnmente aceptadas y los diferentes modelos que se emplean para analizar la información. También se hará una descripción de las diferentes tareas que pueden realizarse con la Minería de Datos como son la clusterización, clasificación entre otras.

CONOCIMIENTOS PREVIOS

El alumno debe contar con conocimientos previos de análisis estadísticos, programación, bases de datos y matemáticas, estos conocimientos son la base para el desarrollo de modelos matemáticos que permiten realizar el análisis de la información en la búsqueda de patrones ocultos e información relevante que ayuden a los procesos de tomas de decisiones dentro de las organizaciones.

5.1 INTRODUCCIÓN

En la actualidad existen grandes cantidades de información relacionada con una amplia gama de temas (ejemplos, consumidores de productos, pacientes, clientes, proveedores, empresas, gobierno, etc.). De acuerdo a la mayoría de las predicciones, un gran porcentaje de la información humana estará disponible en la Web. Estas enormes cantidades de datos plantean un gran desafío, es decir, la búsqueda de una utilidad de toda esta información para que sea más útil [Garofalakis et al., 1999].

Por tal motivo las personas están expuestas a una gran cantidad de información que está disponible para el estudio. Hoy en día también hay una gran cantidad de aplicaciones y servicios que están disponibles a través de Internet y que generan aun más información sobre los usuarios de estas (chats, ventas, etc.), sin embargo, mucha de esa información no es útil para muchas personas, pero en el área de minería de datos, toda la información disponible en las Organizaciones y en el Internet representa una oportunidad de trabajo y es posible obtener información relevante [Ponce et al., 2009].

El Descubrimiento de Conocimiento y Minería de Datos son herramientas poderosas de análisis de datos. El rápido crecimiento que ha tenido estas tecnologías en los últimos años exige un análisis urgente de su impacto social.

Los términos "Descubrimiento de Conocimiento" y "Data Minería" se utilizan para describir la extracción de información no trivial, previamente desconocida e información potencialmente útil de los datos [Wahlstrom y Roddick, 2000], El termino Descubrimiento del Conocimiento es un concepto más amplio que describe el proceso de búsqueda de grandes volúmenes de datos para patrones que se pueden considerar los conocimientos acerca de los datos. La rama más conocida del descubrimiento de conocimiento es la minería de datos.

Alberto muestra una pirámide de la información la cual está conformada por cinco niveles, donde se muestra como los datos se pueden convertir en sabiduría. Los niveles son ruido, datos, información, conocimiento y sabiduría (ver Figura 1).

Figura 5.1: Pirámide de la Información [Hernández, 2013].



El ruido representa la base de la pirámide, este representa cadena de caracteres sin sentido, significado ni orden, cuya naturaleza se desconoce.

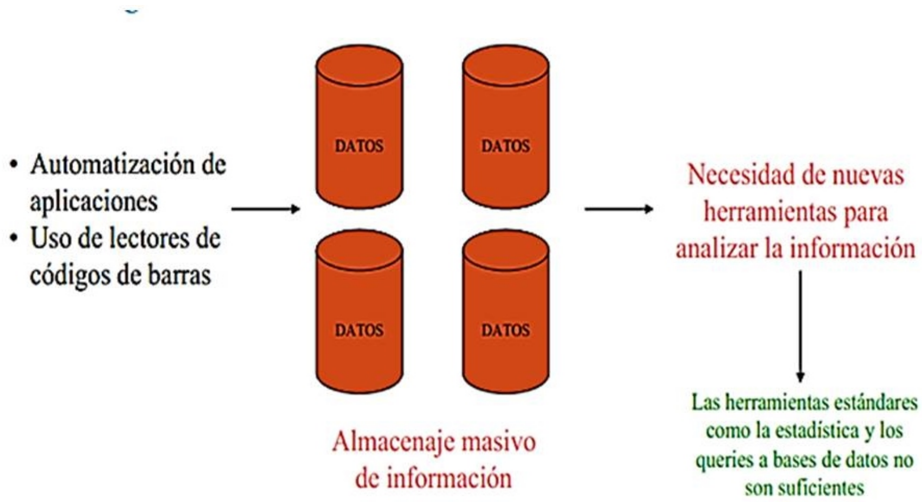
El segundo nivel está representado por los Datos, en este nivel el ruido adquiere un significado, aquí una cadena de caracteres puede significar un nombre, sexo, una dirección y las cadenas de números pueden representar teléfonos, edad, sueldo, costo de un artículo, etc.

La Información la podemos encontrar en el nivel tres de esta pirámide, en este momento los datos adquieren un mayor significado y aparecen también alguna relaciones directas, por ejemplo podemos decir Juan vive en México y su teléfono es 551234567, podemos conocer el costo de un articulo específico, etc.

El cuarto nivel representa la información transformada en conocimiento el cual forma parte esencial para el proceso de la toma de decisiones.

Por último tenemos el nivel de la Sabiduría que representa la cima de la pirámide que también representa el metaconocimiento y lo podemos definir como: "El conjunto de información, hechos y reglas sobre un dominio específico".

La minería de datos nace por la necesidad de nuevas herramientas para analizar la información que día a día se convierte en grandes cantidades de datos para su análisis.



5.2 PREPROCESAMIENTO DE LOS DATOS

Consiste en la selección, limpieza, enriquecimiento, reducción y transformación de datos que puede aplicarse para remover el ruido y corregir inconsistencias.

Para cumplir la transformación de datos se usa la normalización ya que puede aumentar la exactitud y eficiencia de los algoritmos de explotación de datos involucrando la medida de distancias.

Métodos de preprocesamiento

- Limpieza de datos
- Integración y Transformación de datos
- Reducción de datos
- Discretización y generación de jerarquías conceptuales

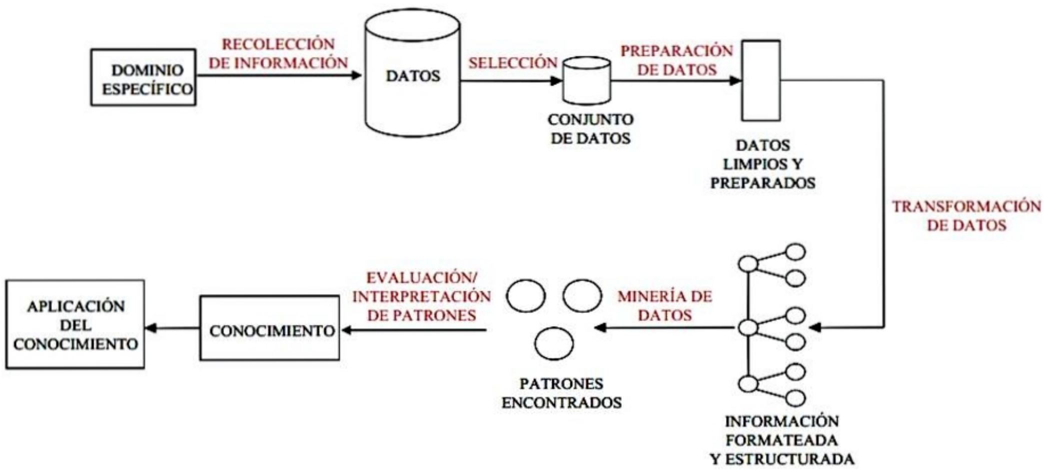
Errores en las BD

- **Datos incompletos**
  - Los atributos de interés pueden no estar siempre disponibles
  - Existen datos que pueden no estar incluidos por ser considerados muy importantes al ser ingresados.
  - Existen datos relevantes que pueden no ser grabados debido a mal entendidos o fallas en los equipos
  - Las modificaciones de la Base de Datos puede ser pasada por alto
  - Los datos faltantes en algunos atributos
- **Datos con ruido**
  - La forma o instrumento de recolección de datos podría ser o estar defectuosa
  - Puede haber errores producidos por el humano o la PC al momento de ingresar datos

- Limitaciones tecnológicas
- **Datos inconsistentes**
    - Datos incompletos pueden resultar debido a inconsistencias en las conversiones de nombres o códigos usados
    - Tuplas duplicadas también requieren limpieza ya que es un error frecuente

Las técnicas de preprocesamiento de datos colaboran al aumento de calidad de los mismos y por lo tanto ayudan a mejorar la exactitud y eficiencia del subsecuente proceso de explotación de datos. De tal manera que un adecuado preprocesamiento de los datos constituye un paso importante en el proceso del descubrimiento del conocimiento en las bases de datos (*Knowledge Discovery in Databases*) por lo que es importante la detección de anomalías en los datos rectificándolos y reduciendo los datos a ser analizados, lo cual puede significar importantes beneficios en la toma de decisiones.

El proceso del descubrimiento de conocimiento en las bases de datos se puede observar en el siguiente diagrama:



Los procesos correspondientes al preprocesamiento de datos son:

- Limpieza de datos
- Integración de datos
- Transformación de datos
- Reducción de datos

Limpieza de datos

Se relaciona con los datos incompletos, con ruido e inconsistentes, por lo cual los procesos de limpieza intentan llenar valores faltantes suavizar el ruido al identificar valores fuera de rango, corregir inconsistencias, también busca las correlaciones ocultas en los datos, identifica los orígenes de datos que son más precisos y determina qué columnas son las más adecuadas para el análisis.

Algunos métodos clásicos para la limpieza de datos son:

- **Para valores faltantes**
    - Ignorar la tupla
    - Llenar el valor manualmente
    - Usar constante global para llenar valores faltantes
    - Usar el valor promedio para completar el valor faltante

  - Usar la media de los atributos para todos los ejemplos pertenecientes a la misma clase de la tupla en cuestión
  - Usar el valor más probable para complotar el valor faltante
- **Datos con ruido**
  - **Encajado(Binning):** consiste en una clasificación, los valores son distribuidos dentro de un numero de paquetes o cajas, los valores de cada caja son reemplazados por el promedio de los datos que contiene
  - **Por Clustering o agrupación:** se hacen grupos a través de valores similares y se detectan los valores fuera de rango o anómalos.
  - Se hace una inspección humana computarizada, se buscan datos.
  - Se puede suavizar los datos ajustándolos a una regresión.
- **Técnica para los datos inconsistentes**
  - Pueden ser corregidas usando referencias externas
  - Se pueden detectar inconsistencias desde la codificación de los programas

Integración de los datos

Esta tarea implica la combinación de datos provenientes de distintas fuentes de información, dichas fuentes pueden incluir BD, archivos planos o cubos de datos. Se deben considerar diversos problemas durante la integración, como es el problema de la redundancia, analizar si existen datos que se refieren a la misma entidad en diferentes fuentes, la inconsistencia de los atributos.

Algunas redundancias pueden ser detectadas mediante un análisis de correlación.

Otro punto importante a considerar es la detección y resolución de valores conflictivos de los datos como son tener valores diferentes para un mismo objeto o entidad dependiendo de la fuente de información, diferencias en la representación o formatos de los atributos, etc.

Transformación de los datos

Muchas veces los datos con transformados para darle una forma apropiada para la explotación de datos, esta tarea involucra las siguientes técnicas:

- **Suavización:** consiste en remover el ruido y se utilizan técnicas como el encajado, agrupamiento y regresión.
- **Agregación:** consiste en aplicar funciones de resumen y de agregación a los datos, normalmente se utiliza para la construcción de cubos de datos, para el análisis con múltiples granularidades.
- **Generalización de los datos:** consiste en tomar datos de bajo nivel o información primitiva y reemplazarlos por un concepto de nivel superior a través del uso de jerarquías conceptuales.
- **Normalización:** el objetivo es escalar los atributos de tal manera que caigan dentro de un pequeño y especifico rango como puede ser de -1 a 0 y de 0 a 1
- **Construcción de atributos:** consiste en insertar nuevos atributos y añadirlo al conjunto actual para facilitar el proceso de explotación de datos.

Reducción de datos

Disminuye el tamaño de los mismos, agregando o eliminando características redundantes o realizando clustering.

Las técnicas de reducción se aplican para obtener una representación reducida del conjunto de datos que es mucho menor en volumen, pero mantiene la integridad de los datos originales.

Las estrategias para la reducción son:

- **Agregado al cubo de datos**

- **Reducción de dimensionalidad:** los atributos poco relevantes o redundantes son removidos.
- **Compresión de datos:** se utilizan mecanismos de codificación para reducir la longitud de los datos
- **Reducción numérica:** consiste en reemplazarlos con representaciones de medidas numéricas más cortas
- **Discretización y generación de jerarquías conceptuales:**
  Se utilizan para reducir el número de valores para un atributo continuo, se realiza dividiendo el rango del atributo en intervalos, las etiquetas de intervalos pueden ser utilizadas para remplazar los valores de los datos actuales. La reducción del número de valores para un atributo tiene un beneficio al momento de utilizar métodos de clasificación basados en arboles de decisión.
  Una jerarquía conceptual nos permite ir ordenando los datos en cada paso de manera que en cuanto más pequeño es el número de valores distintos, mas rápido va a ser el proceso. Para poder realizar una jerarquía conceptual pueden ser aplicadas diversas técnicas de discretización de manera recursiva.
- Encajado
- Análisis de histogramas
- Análisis de cluster
- Discretización: basada en la entropía(grado de conocimiento que se tiene de las cosas)

Reglas de asociación

Las reglas de asociación dentro de la Minería de Datos encuentran asociaciones o correlaciones a través de grandes conjuntos de datos. El descubrimiento de relaciones de interés puede ayudar en muchos procesos de toma de decisión dentro de las organizaciones, tales como diseño de catalogo, marketing, análisis de perdidas, etc.

Un ejemplo típico de reglas de asociación de Minería de Datos es el análisis de la canasta del mercado. Este proceso analiza los hábitos de los compradores encontrando asociaciones entre los diferentes productos de los clientes en base a sus compras.

El descubrimiento de tales asociaciones puede ayudar a vendedores en la estrategia de mercado o hallando cuales productos los clientes compran juntos.

Algunas preguntas asociadas a este tema son:

1. ¿Como es posible hallar estas reglas, para grandes cantidades de datos donde estos son transaccionales o relacionales?
2. ¿Cada regla de asociación es siempre la de mayor interés?
3. ¿Como es posible guiar el proceso de minería para detectar dichas asociaciones?.
4. ¿Que lenguaje de construcción es útil para definir consultas con DM asociadas?

5.3 OLAP-OLTP

OLTP (On-line transaction processing): Bases de Datos orientadas al procesamiento de transacciones donde se define el comportamiento habitual de un entorno operacional de gestión:

1. Altas, bajas, cambios y consultas
2. Consultas rápidas y escuetas
3. Poco volumen de información
4. Transacciones rápidas
5. Gran nivel de concurrencia

OLAP (on-line analytied process): Bases de datos orientadas al procesamiento analítico que determina el comportamiento de un sistema de análisis de datos y elaboración de información:

1. Solo consultas
2. Consultas pesadas y no predecibles
3. Gran volumen de información histórica
4. Operaciones lentas

Características	OLTP	OLAP
Tamaño BD	GB	GB a TB
Origen datos	Interno	Interno y Externo
Actualizacíon	On-line	Batch
Periodo	Actual	Histórico
Consultas	Predecibles	Ad Hoc
Actividad	Operacional	Analítica

5.4 Arquitectura de un Sistema Típico de Minería de Datos

En el inicio del procesamiento de datos el tratamiento de información se realizaba mediante archivos secuenciales donde se almacenaban los datos que se procesarían, lo que a la larga implicaba el tratamiento de archivos completos por los programas de aplicación. Con la necesidad del procesamiento de datos surgió un gran avance en los sistemas computacionales de procesamientos de datos, lo que llevo a que dichos sistemas se hicieran de gran importancia por lo que las empresas comenzaron a reconocer que la información era un recurso corporativo de gran valor por lo que comenzaron a presionar a los sistemas de información para la gestión en cuanto a la utilización de la potencia de la computadora para producir información a partir de los datos corporativos.

Lo que comenzó con la demanda de los sistemas de bases de datos, los que garantizarían más efectivamente el acceso a los datos y su manipulación, cuyo fundamento era una estructura jerárquica de los datos. Permitiendo la recuperación de múltiples registros asociados con un registro único de otro archivo. Inmediatamente después, se desarrollaron los sistemas de base de datos en redes que soportaron interrelaciones entre registros de archivos diferentes mucho más complejas. [7]

En la Figura 5.2 se muestra la arquitectura de un Sistema Típico de Minería de Datos. En donde:

La Base de datos es un repositorio de información como lo son datawarehouse y hoja de cálculo, entre otras.

El servidor de base de datos se utiliza en los sistemas en red para obtener la información en el proceso de minería de datos en el que se encuentre.

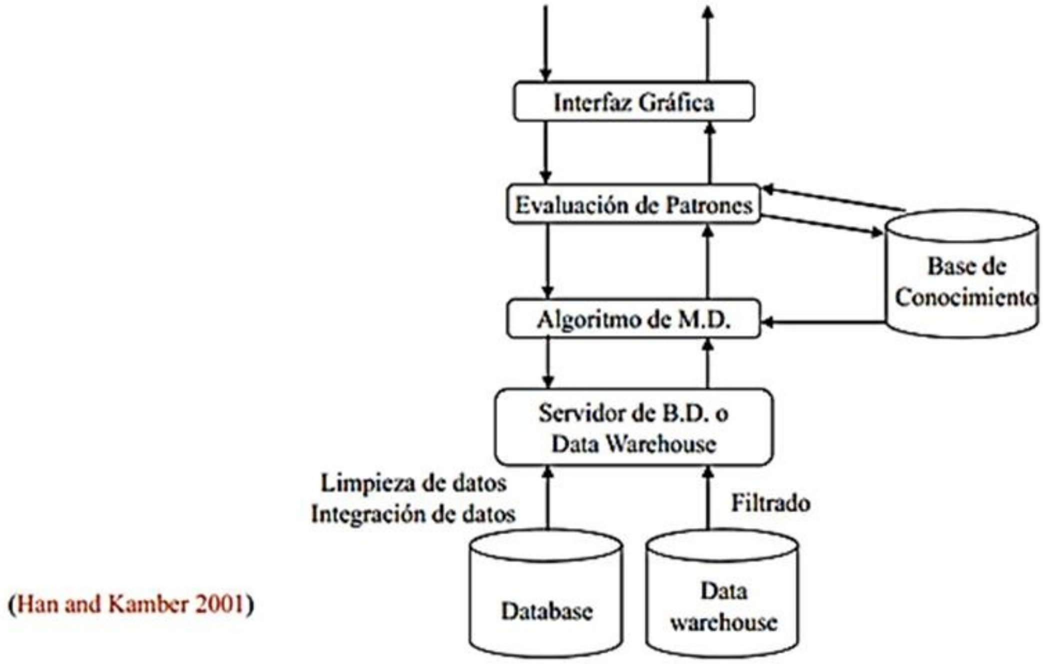
Base de conocimiento: referente al conocimiento del dominio para guiar la búsqueda, evaluar los patrones, meta-datos y obtención de conocimiento previo de los datos

Algoritmo de minería de datos: Permite realizar diferentes tipos de análisis como la caracterización, asociación, clasificación, análisis de grupos, evolución y análisis de desviaciones.

La evaluación de patrones: son medidas para conocer la importancia de los patrones e interactuando con el algoritmo de minería de datos guía la búsqueda hacia patrones interesantes.

Interfaz gráfica: permite la interacción con el usuario, permite la elección de la tarea de minería de daos, provee la información para enfocar la búsqueda, ayuda a evaluar los patrones, explorar los patrones encontrados y la base de datos original y visualizar los patrones en distintas formas

Figura 5.2: arquitectura de un Sistema Típico de Minería de Datos



5.4.1 Arquitectura Centralizada

Los sistemas de arquitectura centralizada se basan en la existencia de una máquina servidora que almacena los datos y las aplicaciones que los procesan, por lo que se ejecutan en un único sistema informático sin interaccionar con ninguna otra computadora.

Ventajas

- Cuentan con un gran nivel de seguridad
- Fácil de administrar

Desventajas

- Representan un alto costo
- El procesamiento de los datos implica que la máquina tenga una sobrecarga de procesos

5.4.2 Arquitectura De Bases de Datos Cliente/Servidor

La arquitectura de Base de datos Cliente/servidor se basa en la existencia de dos tipos de aplicaciones ejecutándose de forma independiente las cuales fungen una como servidor a la que se le realizaran las peticiones y la otra como cliente la que pedirá la información al servidor

Ventajas

- Realiza un reparto de cargas
- Sistema fácilmente escalable

Desventajas

- Requiere un fuerte rediseño de todos los elementos involucrados en los sistemas de información
- Implica un gran gasto en su implantación

Figura 5.3: Arquitectura Centralizada

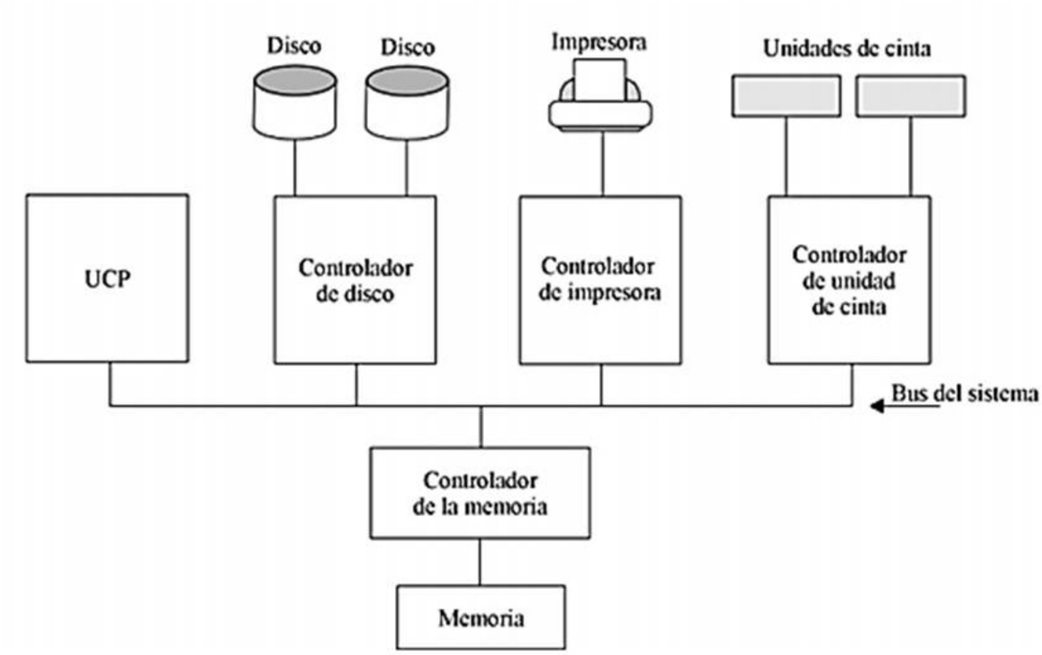
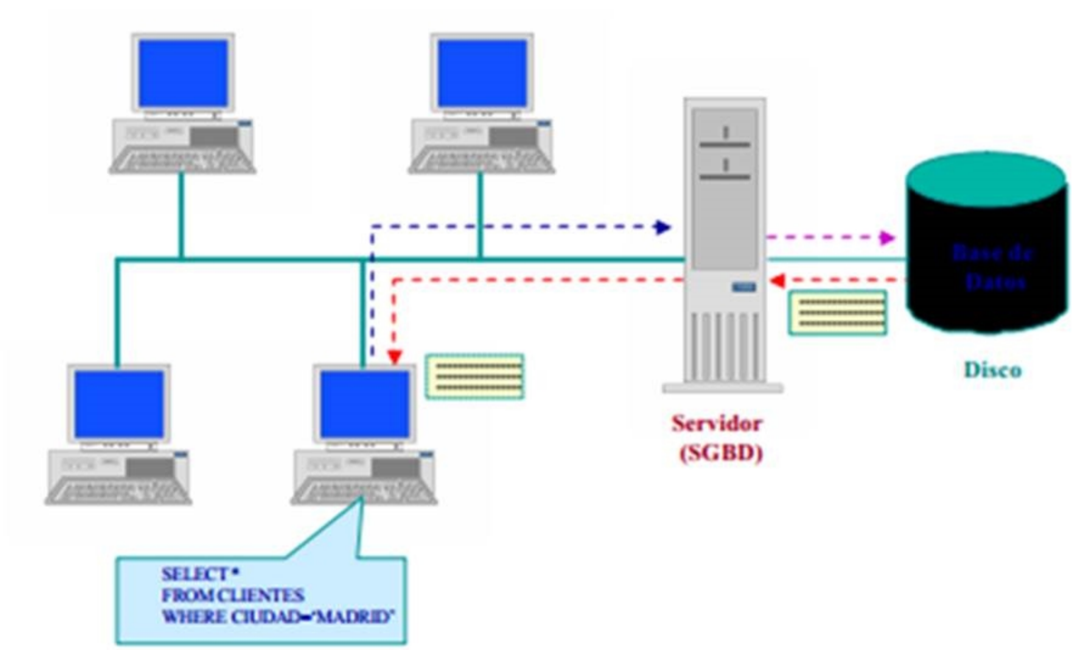


Figura 5.4: Arquitectura De Bases de Datos Cliente/Servidor



5.5 Otras Técnicas Computacionales en Minería de Datos

Las técnicas estadísticas y de investigación operativa son las herramientas básicas que se han empleado en la minería de datos. En los últimos años se han incorporado, tanto a los paquetes estadísticos tradicionales, como a los sistemas de minería de datos unas técnicas estadísticas y algoritmos que resultan especialmente útiles al analizar grandes volúmenes de información. [9]

5.5.1 Sistemas Expertos

Sistemas de la información que simulan el comportamiento de expertos humanos en el proceso de toma de decisiones, para lo que requiere una base de datos de conocimiento para la cual se definen diferentes reglas de comportamiento que regulan el funcionamiento del programa. Cuentan con un motor de inferencia el cual es el encargado de gestionar los datos de entrada, a medida que se usa el sistema experto se incorpora nueva información en la base de datos con el objetivo que el sistema tenga un proceso de aprendizaje.

5.5.2 Lógica Difusa

Método de razonamiento implementado en diversos modelos y productos industriales para representar situaciones en las que los problemas de clasificación están afectados de incertidumbre, es similar al pensamiento humano, que puede procesar información incompleta o incierta, característico de muchos sistemas expertos.

5.6 Aplicaciones de la Minería de Datos

- Recuperación de Información
- Sistemas Médicos
- Análisis de genes y proteínas
- Tráfico
- Hábitos de compra
- Identificación de patrones en recursos humanos
- Comportamiento en Internet
- Juegos
- Terrorismo

Y en diversas áreas de la ciencia e Ingeniería en donde se requiera el procesamiento de la información.

5.7 Herramientas Informáticas

Se han desarrollado diversos programas de minería de datos entre los que podemos encontrar:

Weka: contiene una colección de herramientas de visualización y algoritmos para análisis de datos y modelado predictivo, unidos a una interfaz gráfica de usuario para acceder fácilmente a sus funcionalidades. En un inicio se utilizo para modelar algoritmos implementados en otros lenguajes de programación, más unas utilidades para el procesamiento de datos desarrolladas en C para hacer experimentos de aprendizaje automático.

Clementine: Herramienta de data mining que permite desarrollar modelos predictivos y desplegarlos para mejorar la toma de decisiones. Está diseñada teniendo en cuenta a los usuarios empresariales, de manera que no es preciso ser un experto en data mining.

SAS Analytics / SAS: Suite de soluciones analíticas que permiten transformar todos los datos de la organización en conocimiento, reduciendo la incertidumbre, realizando predicciones fiables y optimizando el desempeño.

RapidMiner / Yale: Es el líder mundial de código abierto para la minería de datos debido a su combinación de su tecnología de primera calidad y su rango de funcionalidad. Esta aplicación de RapidMiner cubre un amplio rango de minería de datos. Además de ser una herramienta

flexible para aprender y explorar la minería de datos, la interfaz gráfica de usuario tiene como objetivo simplificar el uso para las tareas complejas de esta área.

5.8 ACTIVIDADES DE APRENDIZAJE

Los archivos necesarios para el siguiente ejercicio se encuentran en la página:http://repository.seasr.org/Datasets/UCI/arff/

1. Análisis de los datos

1.1. Obtención de los datos
Descargar el siguiente conjunto de datos:

•iris data set: iris.arff

Abrir el archivo de datos con un editor, y estudiar su contenido:

1. ¿Cuántos atributos caracterizan los datos de esta tabla de datos?
2. Si suponemos que queremos predecir el último atributo a partir de los anteriores, ¿estaríamos ante un problema de clasificación o de regresión?

1.2. Estudio estadístico de los datos

- Lanzar la herramienta weka
- Lanzar el Explorer
- Abrir el _chero iris.arff

Una vez cargado el conjunto de datos, en la sección attributes se puede dar click sobre cada atributo para obtener información estadística de ellos. Contestad a las siguientes preguntas:

1. ¿Cuál es el rango de valores del atributo petalwitdth?
2. Con la información que puedes obtener visualmente, ¿qué atributo/s crees que son los que mejor permitirán predecir el atributo class?

1.3. Aplicación de filtros

1. Aplicar el filtro filters/unsupervised/attribute/normalize sobre el conjunto de datos. ¿Qué efecto tiene este filtro?
2. Aplicar el filtro filters /unsupervised/instance/RemovePercentage sobre el conjunto de datos. ¿Qué efecto tiene este filtro?
3. Grabar el conjunto de datos como iris2.ar_.
4. Aplicar el _ filtro filters /unsupervised/attribute/Discretize sobre el conjunto de datos. ¿Qué efecto tiene este filtro?

1.4. Visualización
Volver a cargar el conjunto de datos iris2.arff. Pulsar la pestaña Visualize. Aumentar Point Size a 5 para visualizar los datos mejor.

1. Aumentar el valor de Jitter ¿Qué efecto tiene?

2. Clasificación

El objetivo de este ejercicio es familiarizarse con las primeras técnicas de análisis de datos. En concreto, con los árboles de decisión.

2.1. Clasificador ZeroR

- Cargar el conjunto de datos iris.arff.
- En la pestaña Classify, seleccionar el clasificador ZeroR.
- En las Test Options seleccionar Use trainning set, y pulsar el botón de Start para que genere el clasificador.
- En un instante, en la ventana de salida aparecerán los datos de la clasificación realizada. Analizar esta salida.
    1. ¿Qué modelo genera el clasificador ZeroR?
    2. ¿Cuántas instancias del conjunto de entrenamiento clasifica bien?
    3. ¿Qué porcentaje de instancias clasifica bien?
    4. ¿Qué crees que indica la matriz de confusión?

2.2. Clasificador J48

- Cargar el conjunto de datos iris.arff. En la pestaña Classify, seleccionar el clasificador trees/j48.
- En las Test Options seleccionar Use trainning set, y pulsar el botón de Start para que genere el clasificador.
    1. ¿Cuántas hojas tiene el _árbol generado con J48?
    2. ¿Cuántas instancias del conjunto de entrenamiento clasifica bien?
    3. ¿Qué porcentaje de instancias clasifica bien?
    4. Analizar la matriz de confusión: ¿qué ha clasificado mal?
    5. Pulsar el botón de More Options y seleccionar la opción de Output predictions. ¿En qué instancias se ha equivocado?
    6. Elegir una instancia que J48 haya clasificado erróneamente y a analizar por qué

Además, utiliza alguna de las herramientas de visualización de Weka:

- En la ventana de Result list, pulsa en el botón derecho sobre el modelo generado con J48 para desplegar las opciones. Pulsa sobre Visualize Tree
- En la ventana de Result list, pulsa en el botón derecho sobre el modelo generado con J48 para desplegar las opciones. Pulsa sobre Visualize Errors

2.3. Clasificador ID3

Cargar el conjunto de datos iris.arff. Seleccionar el clasificador ID3 y utilizarlo para generar un árbol de decisión.

1. ¿Has podido ejecutar el algoritmo ID3 sobre el conjunto de datos directamente? ¿Por qué?
2. ¿Qué acciones has llevado a cabo para poder ejecutarlo?
3. ¿Qué porcentaje de éxito sobre el conjunto de entrenamiento has obtenido?
4. ¿Qué porcentaje de éxito obtienes si utilizas como mecanismo de evaluación la validación cruzada?
5. ¿Qué porcentaje de éxito estimas que obtendrás en el futuro sobre nuevos datos con el árbol generado con ID3?

2.4. Árboles de Regresión

Cargar el conjunto de datos cpu.arff. Entre los algoritmos ID3, J48 y M5P, elegir uno de

ellos para aproximar el atributo class sin que sea necesario tratar los datos de entrada de ninguna forma.

1. ¿Qué algoritmo has elegido? ¿Por qué?
2. ¿Qué porcentaje de error obtienes si utilizas como mecanismo de evaluación la validación cruzada?
3. ¿Por qué no disponemos ahora de una matriz de confusión?

3. Agrupación

El objetivo de este ejercicio es familiarizarse con algunas técnicas de agrupación. Para ello, vamos a utilizar también el conjunto de datos iris.arff.

- Cargar el conjunto de datos iris.arff.
- Eliminar el atributo class
- Ejecutar el algoritmo SimpleKMeans para generar 3 grupos. ¿Qué medida de rendimiento genera Weka?
- ¿Qué valor proporciona?
- Ejecutar el algoritmo SimpleKMeans para generar 5 grupos. ¿Cómo mejora la medida de rendimiento?
- Utilizar la herramienta de visualización de grupos para comparar los dos resultados. ¿Puedes obtener alguna conclusión?
- Ejecutar el algoritmo EM con los parámetros por defecto. ¿Cuántas distribuciones genera? ¿Hay alguna relación con alguno de los resultados generados con SimpleKMeans?

4. El Experimenter

El objetivo de este ejercicio es familiarizarse con una herramienta avanzada de análisis de datos integrada en Weka, denominada Experimenter. Esta herramienta permite ejecutar distintos algoritmos de minería de datos sobre distintos conjuntos de datos, de forma que su ejecución secuencial hace más rápida su ejecución, así como la evaluación de los resultados.

Para ello, seguir los siguientes pasos:

- Pulsar el botón New para generar un nuevo experimento
- Seleccionar los conjuntos de datos: iris.ar_, soybean.ar_ y labor.ar_
- Seleccionar los clasificadores: J48,IBK con K = 1, IBK con K = 3, IBK con K = 5, y SVO
- En el apartado Results Destination seleccionar CSV file y utilizar el botón de Browse para elegir el archivo
- Pulsar la pestaña Run y pulsar el botón de Start
- Una vez finalizado el proceso, abrir una hoja de cálculo, y cargar el archivo CSV.
- En ese archivo, se muestra en cada fila los datos de cada ejecución, incluyendo el conjunto de datos, el clasificador utilizado con sus parámetros, así como datos sobre sus resultados
- Localizar la columna que mide el porcentaje de éxito
- Obtener la media del porcentaje de éxito para cada clasificador y conjunto de datos

Una vez realizados los pasos anteriores, responder a las siguientes preguntas:

¿Qué resultados ha obtenido cada clasificador en cada conjunto de datos?

¿Qué algoritmo ha obtenido mejores resultados en cada conjunto de datos?
¿Son los resultados del mejor algoritmo mucho mejores que los del resto?

5.9 MATERIAL DE REFERENCIAS A CONSULTAR * OPCIONAL

[1] msdn.microsoft.com/es-es/library/ms174949.aspx
[2] www.sinnexus.com/business_intelligence/datamining.aspx
[3] www.webmining.cl/2011/01/proceso-de-extraccion-de-conocimiento/
[4] Universidad de Panamá, Universidad Carlos III de Madrid, Gestión y Tecnología del Conocimiento, Minería de datos, Agosto-Septiembre, 2008
[5] Minería de Datos, Teleprocesos y Sistemas Distribuidos, FACENA-UNNE, Octubre, 2003
[6] Pang-Ning Tan, Michael Steinbach, Vipin Kumar , Introduction to Data Mining, PEARSON, Addison Wesley, 2006, Boston, USA, 2006
[7] Óscar González Martín, Francisco Ruiz González, 1999/2000. Arquitecturas De Sistemas de bases de datos, Universidad de Castilla la mancha, Escuela superior de informática Bases de datos.
[8] Jesús Antonio González Bernal, Minería de Datos, Universidad Politécnica de Puebla
[9] José Ma. Caridad Y Ocerin, La Minería de Datos: Análisis de Bases de Datos en la Empresa, 2001, Universidad de Córdoba, Boletín de la Real Academia de Córdoba 141, 357-370

5.10 Bibliografía

Garofalakis, M..; Rastogi, R.; Seshadri, S. & Shim, K. (1999). Data Mining and the Web: Past, Present and Future, Proceedings of 2nd ACM International Workshop on Web Information and Data Management (WIDM), pp. 43-47, Missouri, USA, November 1999, ACM, Kansas City

Hernández J. A. (2013). Generación, Tratamiento y Análisis de Información en las Organizaciones. Juan Pablos Editor. Universidad Autónoma del Estado de Morelos. 174 p. ISBN: 978-607-7771-96-8.

Hernández, J.; Ochoa, A.; Muñoz, J. & Burlak, G. (2006). Detecting cheats in online student assessments using Data Mining, Proceedings of The 2006 International Conference on Data Mining (DMIN'2006), pp. 204-210, Las Vegas, USA, June 2006, Nevada City.

Ponce J., Hernández A., Ochoa A., Padilla F., Padilla A., Álvarez F. and Ponce de León E. (2009). Data Mining in Web Applications. En el libro "Data Mining and Knowledge Discovery in Real Life Applications", Editado por Julio Ponce and Adem Karahoca, ISBN 978-3-902613-53-0, Enero.

Varan, S. (2006). Crime Pattern Detection Using Data Mining, Oracle Corporation.

Wahlstrom K., & Roddick J. (2000). On the Impact of Knowledge Discovery and Data Mining, Proceedings of Australian Institute of Computer Ethics Conference (AiCE2000), Canberra, Australia, April 2000, Sydney City.

6 — Datos Temporales y Datos Espaciales

Elaborado por: Cristina Bender - Claudia Deco
Universidad Nacional de Rosario
Universidad Católica Argentina, Campus Rosario

6.1 Bases de Datos Temporales (*Temporal DataBases*)

6.1.1 Introducción

Una gran variedad de aplicaciones del mundo real manejan datos variables con el tiempo. Por ejemplo: aplicaciones financieras, tales como el manejo de portafolios, los libros contables, los movimientos bancarios, etc.; aplicaciones que requieren mantenimiento de registros, tales como los registros médicos (historias clínicas); el manejo de inventarios, el control de personal, etc.; aplicaciones de programación de reservas para hoteles, trenes, aviones; aplicaciones para la toma de decisión, en los almacenes de datos (*DataWarehouse*); aplicaciones científicas, tales como aquellas para monitoreo del clima, aquellas que requieren en un laboratorio científico la realización de muchos tests, muchas versiones del mismo test, y donde cada test involucra secuencias precisas de tiempo; entre otras. Todas estas aplicaciones requieren almacenar datos junto con referencias temporales. Esto nos lleva a la necesidad del uso de Bases de Datos Temporales.

Las Bases de Datos Temporales están entonces diseñadas para poder capturar información variante en el tiempo a la vez que proporcionan un marco para mantener la historia de los cambios producidos en una fuente de datos.

Los trabajos sobre Bases de Datos Temporales se han centrado en el enriquecimiento con el agregado de características de tiempo de los modelos tradicionales (relacional y orientado a objetos) de las bases de datos y los lenguajes de consulta como el SQL.

En las bases de datos, un *Dato* es una representación codificada de los hechos; en las Bases de Datos Temporales un *Dato Temporal* es una representación codificada de los hechos con el agregado de marcas de tiempo. Esto es, en una Base de Datos Temporal cada hecho registrado tiene una marca de tiempo asociada. Entonces, tomando como base el modelo relacional, en una Base de Datos Temporal Relacional una relación temporal es aquella en que cada tupla contiene al menos una marca de tiempo.

Por ejemplo, sean las tres sentencias siguientes:

1. El proveedor S1 fue contactado (para ser contratado) el 01/07/99.
2. El proveedor S1 fue contratado el 01/07/99.
3. El proveedor S1 fue contratado durante el período comprendido desde el 01/07/99 hasta hoy.

Sea una relación R(S#, Fecha), donde el atributo S# corresponde al código del proveedor, y el atributo Fecha contiene la fecha indicada en la sentencia. Todas las sentencias anteriores

son representables en R con la tupla (“S1”; 01/07/99). Pero las tres sentencias anteriores no son equivalentes. La primera sentencia dice lo que pasó en un momento dado, mientras que las sentencias 2 y 3 sí son equivalentes dado que se refieren a lo que pasó en un período de tiempo.

Las Bases de Datos convencionales manejan bien la información que se produce en un instante dado de tiempo, pero no aquella relacionada con un intervalo de tiempo. Esto es, dados dos hechos *a* y *b*, las Bases de Datos Temporales deben ser capaces de utilizar los operadores de Allen para responder cuestiones como las siguientes: *a* ocurre antes (*before*) que *b*, *a* ocurre al mismo tiempo (*equals*) que *b*; cuando termina *a*, comienza (*meets*) *b*; *a* y *b* se solapan (*a overlaps b*); *a* ocurre durante (*during*) *b*; *a* y *b* comienzan juntos (*a* starts *b*); y, *a* y *b* finalizan juntos (*a* finishes *b*).

En el Apartado 6.4.1. de este capítulo se lista la bibliografía utilizada en este apartado y que es orientativa para profundizar el tema de Bases de Datos Temporales.

6.1.2 El Tiempo en las Bases de Datos Temporales

El tiempo es infinito y continuo. Para representarlo, se transforma en un conjunto discreto, esto es un conjunto de puntos denominados gránulos. Así, la línea del tiempo será una secuencia de puntos de alguna granularidad. Por ejemplo, la sentencia “Martín nació en Nápoles el 10 de Junio de 1986” se puede representar como un Instante, considerando una granularidad de Días. La sentencia “Raquel estuvo en lista de espera de Enero a Octubre” corresponde a un Intervalo con una granularidad de Meses. La sentencia “Ezequiel estudió Letras de 2001 hasta 2004 y Psicología de 2005 hasta 2007” es un Elemento temporal con una granularidad de Años.

Hay dos nociones de tiempo relevantes en una Base de Datos Temporales: el Tiempo Válido y el Tiempo Transaccional. El Tiempo Válido es el momento en el cual un hecho es cierto en la realidad que modela. El Tiempo Transaccional es el momento en el cual un hecho se graba en la base de datos. Estos tiempos no tienen por qué coincidir. Por ejemplo, supongamos que tenemos una base de datos temporal con datos sobre el siglo XVIII; el tiempo válido para estos hechos es algún momento entre 1700 y 1799; y el tiempo transaccionalcomienza cuando se insertan los hechos en la base de datos, por ejemplo 2 de marzo de 2014.

Entonces, el Tiempo Válido (*Valid Time*) es el tiempo en el cual el evento ocurre en la realidad modelada, independientemente de si dicho evento fue registrado o no en una base de datos. Los Tiempos Válidos pueden encontrarse en el pasado, en el presente o en el futuro (si es un hecho que un evento ocurrirá en un cierto momento). Por definición, todos los eventos tienen asociado un Tiempo Válido, pero muchas veces los Tiempos Válidos no necesitan ser registrados en una base de datos; por ejemplo, si son desconocidos o irrelevantes. Y el Tiempo Transaccional (*Transaction Time*) es el tiempo en el cual un evento se hizo presente en una base de datosen la forma de un dato almacenado. Esto es, el Tiempo Transaccional captura los distintos estados de una base de datos.

Los Tiempos Válidos y los Tiempos Transaccionales son ortogonales, esto es, son independientes entre sí. El Tiempo Válido puede ser limitado o ilimitado, pero el Tiempo Transaccional es siempre limitado en ambos extremos, dado que comienza al crearse la base de datos y no supera el momento actual. Sucesivas transacciones pueden mencionar una variedad de Tiempos Válidos (pasados, presentes o futuros) e involucran Tiempos Transaccionales que crecen monótonamente.

Los modelos de datos y los Sistemas de Gestión de Bases de Datos pueden manejar alguno de estos tiempos, ambos o ninguno. Así, las dos nociones de tiempo, tiempo válido y tiempo transaccional permiten la distinción de diferentes formas de Bases de Datos Temporales: una base de datos histórica que almacena datos con respecto a un tiempo válido; una base de datos *rollback* que almacena datos con respecto al tiempo de transacción; una base de datos *snapshot* que almacena sólo un estado del mundo real, usualmente el estado más reciente; o una base de datos bitemporal que almacena datos con respecto a las dos nociones de tiempo, y posee cuatro

atributos temporales (tiempo inicial válido, tiempo final válido, tiempo inicial de transacción, y tiempo final de transacción).

6.1.3 Extensiones al Modelo Relacional

Hay dos formas de extender el modelo relacional, a través de un marcaje de las tuplas o mediante un marcaje de los atributos.

El marcaje de tuplas es la forma más común en el modelo relacional. Indica que la validez de una tupla viene determinada por un atributo temporal. Típicamente se utilizan "desde" y "hasta" para representar los intervalos de tiempo. Una desventaja es que una entidad puede estar representada por varias tuplas, por lo que puede no lograrse una representación 1:1 de la realidad y por esto puede existir información redundante. Otra desventaja es que no permite reflejar la evolución de la identidad de las entidades (al heredar la 1NF del modelo relacional).

El marcaje de atributos incluye la información temporal como parte del atributo. Esto es, tanto el valor de la marca de tiempo como la entrada a la que éste hace referencia se almacenan en el mismo atributo de forma anidada. Esto permite reflejar la evolución de los atributos en el tiempo. Además, se disminuye la redundancia de información. Gracias a los atributos multivaluados se logra una correspondencia 1:1 entre la base de datos y la realidad. Sin embargo, sigue habiendo problemas de actualización; y este es un modelo que no cumple la 1NF por lo que su implementación es más difícil.

6.1.4 El modelo relacional temporal y TSQL

El modelo relacional temporal incorpora la semántica temporal en el modelo relacional. Este modelo utiliza marcaje de tuplas. El lenguaje de consulta temporal se denomina TSQL y es un superconjunto de SQL.

TSQL permite formular consultas relacionadas con la naturaleza especial del tiempo y sus datos asociados. Todas las sentencias SQL son válidas en TSQL y tienen la misma semántica. Se permiten relaciones estáticas y variables en el tiempo. Posee construcciones adicionales al SQL tales como expresiones condicionales usando la cláusula WHEN; expresiones para la recuperación de marcas temporales con y sin cálculos; expresiones para la recuperación de información ordenada temporalmente; expresiones para la especificación del dominio temporal; funciones de grupo modificadas; y expresiones para la especificación de la longitud de un intervalo temporal.

Hay que mencionar que en estas bases de datos puede haber tres razones para que una consulta devuelva un valor nulo. Se puede obtener un resultado nulo en una consulta si el valor de un atributo fue almacenado nulo porque no se conocía en el momento de ser almacenado; o si las condiciones especificadas en una cláusula WHERE o WHEN no se pudieron satisfacer; o si los datos no estaban disponibles en el período de tiempo especificado en la base de datos, por ejemplo debido a discontinuidades en la historia del objeto.

6.1.5 Sistemas de Gestión de Bases de Datos Temporales

Un Sistema de Gestión de Bases de Datos Temporales debe soportar un lenguaje de definición de datos temporales, un sistema de restricciones temporales, un lenguaje de manipulación de datos temporales, y un lenguaje de consultas temporales. Esto es, el DDL (Lenguaje de Definición de Datos) de estos sistemas deberá estar extendido para incluir los tiempos Válido y de Transacción en la definición de una tabla. Por otro lado, su DML (Lenguaje de Manipulación de Datos) deberá ser capaz de insertar, modificar, borrar y consultar datos considerando los tiempos válidos, los tiempos transaccionales y el uso de los operadores de Allen, descriptos previamente.

Se mencionan aquí, a modo de ejemplo, dos implementaciones de Bases de Datos Temporales en un Sistema de Gestión de Bases de Datos Relacionales. El primero es Oracle Workspace

Manager (www.oracle.com) que, al momento del relevamiento realizado para este capítulo, permite manejar versiones actuales, futuras e históricas de los datos en la misma base de datos, y cumple con el estándar TSQL2, que se describe en el próximo párrafo. El segundo es TimeDB (www.timeconsult.com/Software/Software.html) que es una implementación *open source*, desarrollada en Java 1.4, corre como un *front end* de Oracle que acepta sentencias TSQL2 y genera sentencias SQL92.

6.1.6 TSQL2

Richard Snodgrass propuso en 1992 que la comunidad científica de las bases de datos temporales realice extensiones a SQL. Formado un comité, se diseñaron extensiones a la edición 1992 del estándar de SQL (ANSI X3.135.-1992 e ISO/IEC 9075:1992). Estas extensiones, conocidas como TSQL2, se desarrollaron durante 1993. La versión definitiva de la especificación del lenguaje TSQL2 se publicó en septiembre de 1994. Algunas partes de TSQL2 se incorporaron a un subestándar de SQL3 (de 1999) llamado SQL/Temporal (ISO/IEC 9075-7). Pero, el proyecto ISO responsable para el soporte temporal fue cancelado a fines de 2001. La especificación TSQL2 incluye ideas y conceptos tales como Tiempo Válido, Tiempo Transaccional, y Tablas Bitemporales. En 2002, Chris Date, Hugh Darwen y Nikos Lorentzo presentan su libro *Temporal Data & the Relational Model* donde tratan varios de estos términos introducidos por TSQL2 y también introducen formas normales que permiten resolver algunos de los problemas.

Así, TSQL2 (`digital.cs.usu.edu/{~}cdyreson/pub/temporal/tsql2.htm`) es un lenguaje de consulta temporal consensuado por un comité de grupos líderes de investigación en bases de datos temporales, y es una extensión de SQL-92. Una de las características de TSQL2 es que incluye cuatro tipos de marcas de tiempo válido: intervalos, instantes, períodos y elementos. Asociadas a estas marcas de tiempo existen tres categorías de operadores relacionados con la selección: extractores, constructores y comparadores. Los extractores de eventos son los operadores BEGIN y END que pueden aplicarse a eventos, períodos y elementos. Los extractores de períodos son FIRST y LAST. Los constructores de eventos son (también) FIRST y LAST. Los constructores de períodos son PERIOD e INTERSECT. Los comparadores de elementos son cuatro: PRECEDES, =, OVERLAPS y CONTAINS. El comparador de períodos es MEETS. Todos los operadores de Allen pueden ser simulados mediante los comparadores de TSQL2.

TSQL2 tiene una variable, indicada con NOW, cuyo valor es el momento actual y es especificado al momento de una consulta o de una actualización. El tiempo asignado a esta variable depende de cuándo se evalúa la consulta o la actualización, esto es el tiempo de referencia. Otros nombres para esta variable son *until changed*, *forever* que corresponde a la mayor marca de tiempo representable (en TSQL2 aproximadamente 18 billones de años a partir del momento actual); y *begining* que es similar a *forever*, pero representa el valor menos infinito.

TSQL2 soporta tiempos transaccionales, que son siempre crecientes. Esto ocurre porque todas las actualizaciones lógicas, incluyendo las eliminaciones, se transforman en inserciones a nivel físico. Esto contrasta con los lenguajes no temporales (que manejan *snapshots*) donde las eliminaciones lógicas y físicas son equivalentes. Si alguna aplicación desea eliminar físicamente datos antiguos, se tiene la operación de “aspirado” (*vacuuming*). Así, toda la información con tiempo transaccional anterior a la fecha de corte indicada en la sentencia será eliminada.

Actividad para el alumno:

- Realizar un relevamiento actualizado de Sistemas de Gestión de Bases de Datos, comerciales y *open source*, que incluyan extensiones temporales. Analizar los sistemas hallados.
- En los sistemas hallados, crear una base de datos que necesite manejar el tiempo y evaluar si cumplen con los operadores de Allen.

6.2 Bases de Datos Espaciales *(Spatial DataBases)*

6.2.1 Introducción

Algunas aplicaciones en las que se utilizan bases de datos espaciales son las imágenes satelitales, que son el ejemplo más claro de datos espaciales; un modelo de circuitos integrados de gran tamaño (*VLSI*); los Sistemas de Información Geográfica (*SIG*s), donde los datos son una red de caminos y lugares de interés, y se utiliza para, por ejemplo el manejo de direcciones; sistemas ambientales, donde los datos pueden corresponder a superficies de tierra, clima, lluvia y fuegos forestales, y se utilizan para, por ejemplo, calcular el caudal de lluvias precipitadas; o, sistemas para la toma de decisiones corporativas y sistemas de soporte empresarial, donde los datos pueden ser, por ejemplo la ubicación de sucursales y los clientes, y se pueden utilizar para, por ejemplo, determinar la mejor ubicación para una nueva sucursal.

Para estas aplicaciones se necesita representar **Objetos** en el espacio, y el **Espacio** propiamente dicho. Con Objetos se hace referencia a distintas entidades ubicadas en el espacio, cada una con su propia descripción geométrica. De esta forma se pueden modelar ciudades, ríos, bosques., etc. Para representar el Espacio hay que caracterizar cada punto en el espacio. Así, se pueden modelar mapas temáticos que indican, por ejemplo, la división de un país en provincias o la de una ciudad en distritos. Entonces, las Bases de Datos Espaciales registran esta información mediante ciertas abstracciones.

Las abstracciones fundamentales para modelar objetos únicos (o individuales) son punto, línea, y región. Los puntos representan un objeto para el cual sólo su ubicación (y no su extensión) en el espacio es relevante. Por ejemplo, para un área geográfica muy extensa, una ciudad puede representarse como un punto. Las líneas representan conexiones en el espacio. Por ejemplo, una ruta, un río o un tendido eléctrico. Las regiones representan objetos con extensión en dos dimensiones. Puede contener orificios y estar formada por piezas disjuntas. Por ejemplo, una región puede representar un país, un lago o un parque nacional. Los conjuntos de objetos relacionados espacialmente se modelan con particiones o redes. Una partición es un conjunto de objetos de tipo región que se requiere que sea disjunto. Un caso particular es la *relación adyacente*, donde las regiones tienen una frontera común. Una red (*network*) consiste de un conjunto de objetos punto que forman los nodos de un grafo y un conjunto de objetos que forman los contornos que unen dichos nodos. Puede verse como un grafo dentro del plano. En la figura 6.1 se muestra un bosquejo de estas abstracciones.

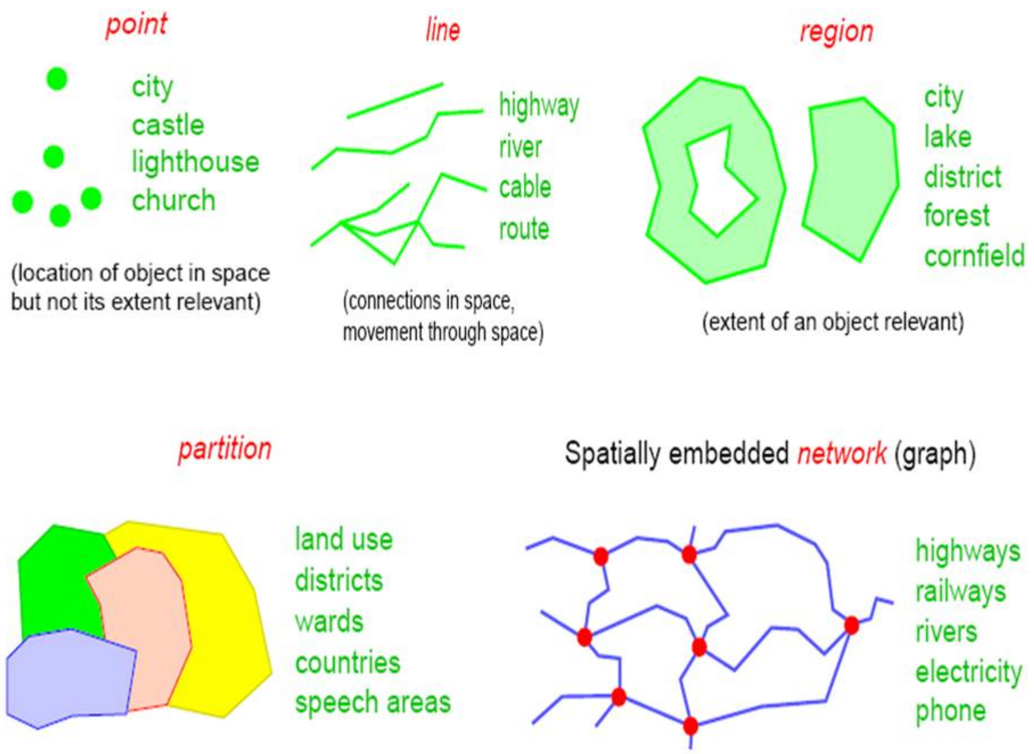
Básicamente existen dos clases distintas de Bases de Datos Espaciales que se diferencian de acuerdo a la forma de obtener y manejar la información. Estas son Bases de Datos de Imágenes, y Bases de Datos Espaciales. En las primeras, los datos se capturan como imágenes digitales y contienen, por ejemplo, imágenes satelitales, tomografías computadas, etc. Las bases de datos de ***imágenes*** permiten manejar fotos ***del*** espacio. El segundo tipo contiene conjuntos de objetos en el espacio. Cada objeto tiene identidad, propiedades, localización y se relaciona con otros objetos del espacio. Entonces, las bases de datos***espaciale***s permiten manejar ***objetos en el*** espacio. Estas últimas son las que se describen en este capítulo.

En el Apartado 6.4.2. de este capítulo se lista la bibliografía utilizada en este apartado y que es orientativa para profundizar el tema de Bases de Datos Espaciales.

6.2.2 Sistemas de gestión de Bases de Datos Espaciales

La incorporación de extensiones para el manejo de datos espaciales radica en la necesidad de manejar de forma integrada tanto los datos descriptivos como los datos espaciales o geométricos de los objetos que la forman. Los sistemas de gestión de bases de datos relacionales sirven para datos no espaciales. Pero no son óptimos para consultas espaciales, tales como “Localizar todas las ciudades a no más de 100 Km del río Paraná, y de más de 100.000 habitantes”, u “Obtener la ciudad más cercana a un punto dado”. Aquí, se trata con dos tipos de datos: espaciales y no

Figura 6.1: Abstracciones fundamentales para modelar objetos



espaciales. Los datos espaciales corresponden a puntos discretos en el espacio, y un espacio ocupado por fenómenos continuos que tienen extensión en dos dimensiones. Los datos no espaciales pueden corresponder a, por ejemplo, nombres de regiones, límite de velocidad en una carretera, cantidad de pobladores de una población, etc.

Una base de datos espacial debe permitir la descripción de los objetos espaciales mediante tres características básicas. La primera es describir sus **Atributos**, esto es saber qué es un objeto mediante la representación de sus características. La segunda es describir su **Localización**, esto es saber dónde está el objeto y qué espacio ocupa. La tercera corresponde a la descripción de su **Topología**, esto es mejorar la interpretación semántica del contexto y establecer jerarquías de elementos a través de sus relaciones.

Un sistema de gestión de bases de datos espaciales (SDBS, del inglés *Spatial DataBase System*) es un sistema de base de datos donde la información espacial o geométrica está ligada a la información alfanumérica, esto es los datos que maneja cualquier base de datos. Pero ofrece tipos de datos espaciales (punto, línea y región) en su modelo de datos y en su lenguaje de consulta, soporta tipos de datos espaciales en su implementación y provee indexación espacial y algoritmos para el *join* espacial. La característica común a todos estos sistemas de gestión es la capacidad relativamente simple de manejar grandes colecciones de objetos geométricos.

Entre los sistemas comerciales se pueden nombrar los siguientes: Oracle Spatial (`www.oracle.com`); ESRI (www.esri.com), que es un software GIS de mapeo orientado al apoyo a la toma de decisión; Intergraph (`www.intergraph.com/`), que provee software de aplicaciones de ingeniería potenciado con aspectos geoespaciales; Galdos Systems (`www.galdosinc.com`), que es un desarrollo de sistemas de tiempo real que se integran, validan y distribuyen datos geoespaciales; y, Microsoft (`www.microsoft.com`), donde en su producto SQL Server 2008 ya soporta datos espaciales.

Entre las ofertas *open source* se pueden mencionar: PostGIS (postgis.refractions.net), que es el soporte para objetos geométricos para PostgreSQL; MySQL (www.mysql.com), que ya en su versión 5.0 agrega soporte para datos espaciales; y, GraphDB (github.com/sones/sones/tree/master/GraphDB/). Además, existe el Consorcio OGC (*Open Geospatial Consortium*) (www.opengeospatial.org) que proporciona estándares OpenGIS, por ejemplo, el modelo para *Geography Markup Language* (GML), que es una gramática XML para expresar características geográficas.

Actividad para el alumno: Realizar un relevamiento actualizado de Sistemas de Gestión de Bases de Datos Espaciales, comerciales y *open source*. Analizar los sistemas hallados.

6.2.3 Operaciones espaciales

Se tienen distintos tipos de operaciones espaciales, aquellas que devuelvan valores booleanos, aquellas que devuelvan valores numéricos, aquellas que devuelvan nuevos objetos espaciales, y aquellas que se realizan en colecciones de objetos espacialmente relacionados. Las operaciones que devuelven valores booleanos, corresponden a relaciones topológicas (igual, disjunto, adyacente, intersección, cubre, contiene, fuera, etc.), a orden espacial (detrás, en_frente, debajo, por_sobre, etc.), y a relaciones direccionales (norte, sur, este, noreste, etc.). Algunas de las operaciones que devuelven valores numéricos son: área, perímetro, diámetro, distancia, distancia máxima, y distancia mínima, entre otras. Las operaciones que devuelven nuevos objetos espaciales, se pueden subclasificar en operaciones de construcción (tales como unión, intersección, diferencia, centro, borde, etc.) y operaciones de transformación (tales como extender, rotar, trasladar, etc.).

Las operaciones que se realizan en colecciones de objetos espacialmente relacionados se subclasifican en operaciones generales (por ejemplo, más cercano, componer, descomponer, etc.); operaciones para particiones (como ser fusión, superimposición, cubrir, etc.), y operaciones para redes (por ejemplo, encontrar el camino_mas_corto entre dos puntos).

Según OpenGIS, la jerarquía de clases geométricas es la siguiente:

- Geometry (abstracta)
  - Point
  - Curve (abstracta)
    - LineString
      - Line
      - LinearRing
  - Surface (abstracta)
    - Polygon
  - GeometryCollection
    - MultiPoint
    - MultiCurve (abstracta)
      - MultiLineString
    - MultiSurface (abstracta)
      - MultiPolygon

La clase **Geometry** es no instanciable, pero tiene propiedades comunes a todos los objetos geométricos. Estas propiedades son **Tipo**, es decir a qué clase instanciable pertenece; **SRID**, o sea el sistema de referencia que describe el espacio de coordenadas para el objeto (generalmente es el espacio euclídeo); **Coordenadas**, que corresponden a números de doble precisión (x,y); **Interior**, **Exterior** y **Límite**, esto es el espacio ocupado por la geometría, todo el espacio no ocupado por la geometría y la interfaz entre ambos; **MBR**, que corresponde al mínimo rectángulo que contiene la geometría; y **Dimensión**, que vale -1 para una geometría vacía, 0 para un punto,

1 para una línea, y 2 para una región.

Luego, cada clase instanciable presenta propiedades particulares. Por ejemplo, para la clase ***Point***, se provee el valor de la coordenada X, valor de la coordenada Y, y el límite es el conjunto vacío; para la clase ***LineString*** se proveen los valores de las coordenadas de segmentos, si es tipo *Line* se dan sólo dos puntos, si es tipo *LinearRing* indica que es simple y cerrada; la clase ***Polygon*** puede tener agujeros, su límite es un conjunto de *LineString*.

6.2.4 Formatos de datos espaciales

Para intercambiar datos es necesario un estándar de representación. En los datos espaciales existen dos estándares para representar objetos geométricos en las consultas: el Formato **WKT** (*Well Known Text*) y el Formato **WKB** (*Well Known Binary*).

El formato WKT está diseñado para intercambiar datos en tipo ASCII. Por ejemplo: un punto se representa con sus coordenadas como POINT(15 20); una línea con cuatro puntos se representa con las coordenadas de esos cuatro puntos en la forma LINESTRING(0 0, 10 10, 20 25, 50 60); y un polígono con un anillo exterior y un anillo interior se representa de la forma siguiente POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7, 5 5)).

El formato WKB está diseñado para intercambiar datos como cadenas binarias. Utiliza enteros sin signo de 1 byte, enteros sin signo de 4 bytes, y números de 8 bytes de doble precisión (según el formato IEEE 754). Por ejemplo, el punto (1,1) se representa con una secuencia de 21 bytes cada uno representado por dos dígitos hexadecimales, que puede descomponerse en la forma siguiente:

Orden de byte	Tipo WKB	Coordenada X	Coordenada Y
01	01000000	0000000000F03F	0000000000F03F

Actividad para el alumno: A partir del relevamiento de Sistemas de Gestión de Bases de Datos Espaciales, comerciales y *open source*, realizado en la Actividad anterior, analizar en cada uno de los sistemas hallados cuál es el estándar de representación que utilizan para el intercambio de datos y cuál es la forma de representación interna de los datos espaciales.

6.2.5 Extensiones espaciales en SQL. El caso de MySQL

Las extensiones espaciales permiten la generación, el almacenamiento y el análisis de elementos geográficos. Un entorno SQL con tipos geométricos es un entorno en el cual al menos una columna de una relación contiene datos de tipo geométrico. En particular, y a modo de ejemplo, se describen brevemente algunas extensiones implementadas por MySQL siguiendo las normativas del consorcio OGC.

La sentencia para la creación de tablas con columnas espaciales, es la conocida en SQL, excepto que las columnas son de un tipo espacial. Por ejemplo, la sentencia de creación de la tabla Geometrica con tres columnas, es la siguiente:

```
CREATE TABLE Geometrica (
           Geom           GEOMETRY,
           Origen_x            POINT,
           Origen_y            POINT);
```

Para cargar datos en los atributos geométricos, sus valores deben ser almacenados en el formato interno, esto es, es necesario convertir los datos desde el formato de intercambio (WKT o WKB) al formato interno del sistema de gestión. También, los valores almacenados pueden extraerse de 3 maneras diferentes, todas usando SELECT: en formato interno, en formato WKT, en formato WKB.

Las operaciones que provee MySQL se agrupan en funciones que:

- convierten las geometrías a diversos formatos: *AsBinary*(g), *AsText*(g), *GeomFromBinary*(wkb), *GeomFromText*(wkt).
- proveen acceso a las propiedades cuantitativas o cualitativas de una geometría. Estas funciones toman como argumento un elemento geométrico y devuelven como resultado una propiedad del mismo. Pueden ser funciones genéricas que se aplican a cualquier geometría, o funciones particulares que corresponden a punto (*Point*), línea (*LineString*) y área (*Polygon*).
- describen relaciones entre dos geometrías. Los estándares definen funciones que prueban relaciones espaciales entre cualquier clase de objetos geométricos, pero MySQL sólo implementa funciones aplicadas a MBR (*Minimal Bounding Rectangles*). Entre las funciones de relaciones espaciales entre los rectángulos que circunscriben a las geometrías están las siguientes: un rectángulo contiene al otro, los rectángulos son disjuntos, los rectángulos son iguales, los rectángulos se intersecan, los rectángulos se solapan, los rectángulos se tocan, y el primer rectángulo se encuentra dentro del segundo rectángulo. El resultado de todas estas funciones es booleano (1 por verdadero, 0 por falso).
- crean nuevas geometrías desde otras ya existentes. Entre estas, se tienen funciones que retornan el mínimo rectángulo que circunscribe el valor geométrico *g*; retornan el punto inicial o el punto final de una línea; devuelven una geometría que representa todos los puntos cuya distancia a la geometría de partida sea menor que un dado valor; retornan geometrías cuyo valor es la resta, la intersección o la unión entre dos geometrías dadas.

Actividad para el alumno: Utilizando MySQL, crear una base de datos con datos espaciales, generar una instancia y evaluar los resultados obtenidos al realizar consultas.

6.3 Bases de Datos Espacio Temporales (*Spatio-Temporal Databases)*

6.3.1 Introducción

Un Sistema de Bases de Datos Espacio-Temporales, es un sistema de bases de datos cuyos objetos tienen una geometría que cambia a lo largo del tiempo. Es decir, son sistemas que tienen la capacidad de gestionar geometrías en cambio continuo. En los últimos años ha habido mucho trabajo en el área de Bases de Datos Espacio-Temporales, también conocidas como bases de datos de objetos móviles. Los campos de aplicación incluyen: aplicaciones demográficas, por ejemplo para evaluar el desplazamiento de poblaciones; aplicaciones vinculadas a la ecología, por ejemplo para analizar el crecimiento / decrecimiento de bosques; aplicaciones de marketing para, por ejemplo, analizar el movimiento de vendedores; aplicaciones para analizar fenómenos naturales, por ejemplo el movimiento de huracanes; aplicaciones en el campo militar, por ejemplo para evaluar el movimiento de tropas, o analizar regiones ocupadas; aplicaciones urbanísticas, por ejemplo para analizar el crecimiento / decrecimiento de ciudades; entre otros.

Si se considera una aplicación de monitoreo de tráfico, un sistema que contenga datos temporales y datos espaciales podrá responder cuestiones tales como: cantidad de vehículos que hay en un área de una ciudad, existencia de congestiones de tráfico en alguna calle, informar a la policía en caso de que ocurra un accidente, asegurar que no hay más aviones en una ruta dada, etc. Estas aplicaciones requieren manejar objetos cuya posición espacial y/o forma cambia en distintos instantes de tiempo. Entonces, es necesario que un Sistema de Bases de Datos Espacio-Temporales sea capaz de representar modelos muy cercanos al mundo real, con todo el dinamismo que esto implica y administrar objetos que cambian su ubicación y/o forma a través del tiempo.

Un objeto espacio-temporal tiene un identificador o clave primaria, una localización y forma

espacial, alguna propiedad temporal y características que lo describen. Los modelos de tiempo y espacio pueden ser discretos o continuos, pero en cualquiera de estas dos alternativas dos objetos no pueden estar en el mismo punto en el espacio y en el tiempo.En el Apartado 4.3. de este capítulo se lista la bibliografía utilizada en este apartado y que es orientativa para profundizar el tema de Bases de Datos Espacio Temporales.

6.3.2 Consultas sobre Datos Espacio Temporales

Respecto a las consultas sobre este tipo de datos, se tienen distintas perspectivas. En los ejemplos siguientes se podrán apreciar las diferencias.
Ejemplo 1: Reportar continuamente la cantidad de autos en una autopista. El tipo de esta consulta es "consulta de rango", El tiempo es el "presente", la duración es "continua", el que consulta está "estático", y el objeto de consulta está "en movimiento".
Ejemplo 2: ¿Cuáles son las estaciones de servicio más cercanas para la próxima hora? La consulta es de tipo "*nearest-neighbor*", el tiempo es "futuro", la duración es "continua", el que consulta está "en movimiento", y el objeto de consulta está "estático".
Ejemplo 3: Enviar un mensaje a los autos informándoles que soy la estación de servicio más próxima. La consulta es de tipo "*nearest-neighbor* inversa", el tiempo es "presente", la duración es "*snapshot*", el que consulta está "estático", y el objeto de consulta está "en movimiento".
Ejemplo 4: Cuál fue la menor distancia entre el banco A y yo en el día de ayer?. La consulta es de tipo "consulta de punto más cercano", el tiempo es "pasado", la duración es "*snapshot*", el que consulta está "en movimiento", y el objeto de consulta está "en movimiento".

Un sistema de bases de datos espacio-temporales deberá ser capaz de responder estos tipos de consulta.

6.3.3 Extensión Espacio-Temporal del Modelo Entidad-Relación

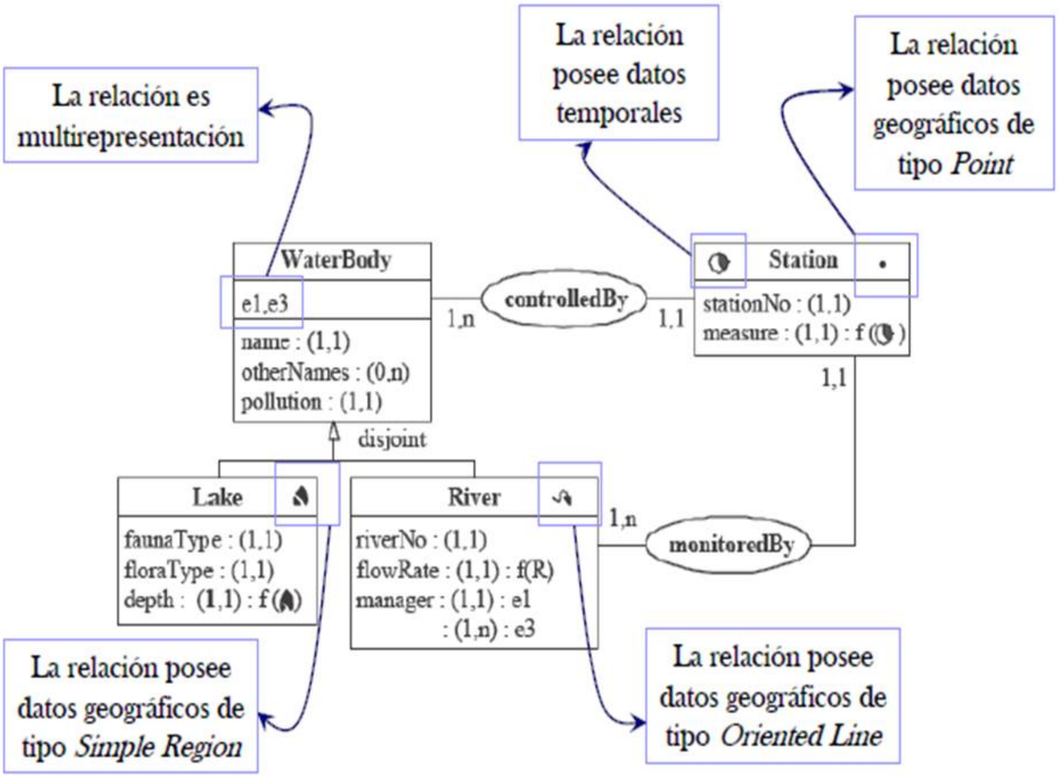
MADS es un modelo conceptual basado en una extensión del modelo Entidad-Relación [Parent et al., 1999]. Incluye varias dimensiones de modelado: estructural, espacial, temporal y multirepresentación. Todas estas dimensiones son ortogonales, es decir que pueden ser incorporadas libre e independientemente a los esquemas.

Las dimensiones espacial y temporal se organizan a partir de una serie de tipos de datos dispuestos jerárquicamente. A nivel espacial partiendo de una clase genérica *Geo*, existen *Point*, *Line*, *Oriented Line*, *Simple Region*, *Point Set*, *Line Set*, entre otras. A nivel temporallos datos pueden variar continuamente o discretamente. Se asocia a cada objeto un ciclo de vida, tomando alguno de los valores *not-yet-existing*, *active*, *suspended* o *disabled*. Los tipos de datos asociados a los objetos temporales son *Temporal* (genérico), *Instan*t, *Interv*al, *Instant Set* e *Interval Se*t. Por ejemplo, el esquema MADS de una aplicación para monitoreo de fuentes de agua se visualiza en la figura 6.2.

En este modelado de requerimientos, se debe representar objetos con posición en el espacio y el tiempo; capturar cambios continuos o discretos de posición en el espacio a lo largo del tiempo; definir atributos del espacio y la organización de ellos entre capas y campos; capturar el cambio de atributos espaciales a lo largo del tiempo; conectar atributos espaciales a objetos; representar relaciones espaciales entre objetos en el tiempo y representar relaciones entre atributos espaciales en el tiempo; especificar restricciones de integridad espacio-temporal, impuestas por el usuario o por el diseñador de la base de datos.

Sean los objetos geométricos A y B, y el tiempo T. Los operadores espaciales que pueden usarse son AREA(A), LENGTH(A), DISJOINT(A,B). Los operadores temporales son BEGIN(A), END(A), T-BEFORE(A,B), INTERVAL (tiempo inicial, tiempo final). Respecto a los operadores espacio-temporales se tienen los siguientes: un operador de ubicación-temporal ST_SP(A,T), que devuelve la representación espacial del objeto A válido en el tiempo T; un operador de

Figura 6.2: Esquema MADS de una aplicación para monitoreo de fuentes de agua



orientación-temporal, que devuelve un valor booleano indicando si existe una relación específica entre dos objetos A y B; un operador métrico-temporal que considera el objeto métrico A en el tiempo T, ST_AREA(A,T), y la distancia entre dos componentes espaciales A y B en el tiempo T, ST_DISTANCE(A,B,T); y un operador topológico-temporal, que devuelve un booleano indicando la relación topológica entre A y B durante el tiempo T, ST_DISJOINT(A,B).

Actividad para el alumno: Realizar un relevamiento de Sistemas de Gestión de Bases de Datos Espacio Temporales, comerciales y *open source*. Analizar los sistemas hallados.

6.3.4 SQLST

SQLST es un lenguaje para el manejo de información espacio-temporal. Es una extensión del SQL estándar, esto es preserva su estructura y sólo añade tipos de datos temporales (DAY), espaciales (POINT, LINE y REGION), y operadores (AREA, OVERLAP, MOVING_DISTANCE). El tiempo se representa discretamente, y los cambios de las geometrías, ya sea en forma o posición, son discretos.

A modo de ejemplo, consideremos un escenario correspondiente a incendios forestales. En este escenario se maneja información sobre bosques, incendios y bomberos. Se necesitan así las relaciones bosque, incendio y bombero:

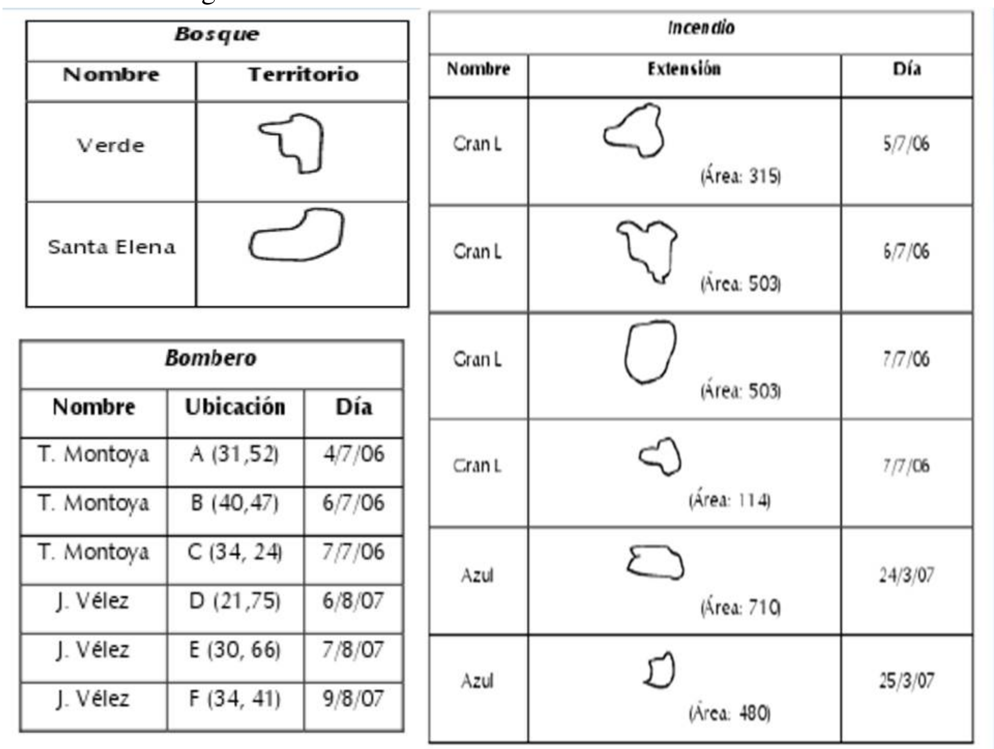
CREATE TABLE **bosque**
(nombre CHAR(30),
territorio REGION);

CREATE TABLE **incendio**
(nombre CHAR(30),
extensión REGION,
día DAY)

CREATE TABLE **bombero**
(nombre CHAR(30),
ubicación POINT,
día DAY)

En la figura 6.3, se presenta una instancia con datos correspondientes a las tres relaciones creadas

Figura 6.3: Instancia con datos de las relaciones creadas



Bosque

Nombre	Territorio
Verde	
Santa Elena	

Bombero

Nombre	Ubicación	Día
T. Montoya	A (31,52)	4/7/06
T. Montoya	B (40,47)	6/7/06
T. Montoya	C (34, 24)	7/7/06
J. Vélez	D (21,75)	6/8/07
J. Vélez	E (30, 66)	7/8/07
J. Vélez	F (34, 41)	9/8/07

Incendio

Nombre	Extensión	Día
Gran L	(Área: 315)	5/7/06
Gran L	(Área: 503)	6/7/06
Gran L	(Área: 503)	7/7/06
Gran L	(Área: 114)	7/7/06
Azul	(Área: 710)	24/3/07
Azul	(Área: 480)	25/3/07

Si se desea saber, por ejemplo, ¿Cuándo y dónde alcanzó el incendio “Gran L” su máxima extensión? , una expresión para la consulta es:

```
SELECT F1.dia, F2.extension, AREA(F1.extension)
FROM incendio AS F1 F2
WHERE F1.nombre="Gran L" AND F2.nombre="Gran L" AND F1.dia=F2.dia
GROUP BY F1.dia
HAVING AREA(F1.extension) =
            (SELECT MAX(AREA(extension))  FROM incendio
              WHERE nombre="Gran L");
```

cuyo resultado se indica en la figura 6.4

Figura 6.4: Resultado de Consulta

F1.dia	F2.extension	AREA
6/7/06		503
7/7/06		503

En la consulta anterior, se pueden evitar el *self-join* y las cláusulas GROUP BY y HAVING, escribiendo la consulta de la forma siguiente:

```
SELECT dia, extension, AREA(extension)
      FROM incendio
      WHERE nombre="Gran L"
            AND AREA(extension) =
                        (SELECT MAX(AREA(extension))
                        FROM incendio
                        WHERE nombre="Gran L");
```

Actividad para el alumno: Utilizando alguno de los sistemas encontrados en la Actividad anterior crear una base de datos con datos temporales y datos espaciales, y generar una instancia. A partir de un conjunto de consultas que necesiten utilizar ambos tipos de datos, analizar las respuestas obtenidas.

6.4 Bibliografía

6.4.1 Bases de Datos Temporales

[Allen, 1983] Allen, Maintaining Knowledge about Temporal Intervals. Communications of the Association of Computing Machinery, 26, No. 11, Nov. 1983.

[Celko, 2005] Celko J. Joe Celko's SQL for Smarties: Advanced SQL Programming. The Morgan Kaufmann Series in Data Management; Morgan Kaufmann; 3rd edition. 2005.

[Date et al., 2002] Date C.J., Darwen H., Lorentzos N. Temporal Data & the Relational Model, First Edition. The Morgan Kaufmann Series in Data Management Systems; Morgan Kaufmann; 1st edition. 2002.

[Jensen and Sondgrass, 1996] Jensen and Sondgrass, Semantics of Time-Varying Information. Information Systems, 21, 1996.

[Sarda, 1990] Sarda, Extensions to SQL for Historical Databases. IEEE Transactions on Knowledge and Data Engeneering, 2, No. 2, June 1990.

[Snodgrass, 1986] Snodgrass R.T. Temporal Databases. IEEE Computer, 19, No. 9, Sep. 1986.

[Snodgrass, 1995] Snodgrass R.T. TSQL2 Temporal Query Language. Computer Science Department of the University of Arizona. Disponible en www.cs.arizona.edu/~rts/tsql2.html.

[Snodgrass, 1999] Snodgrass R.T. Developing Time-Oriented Database Applications in SQL. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann. 1999.

[Wiederhold et al., 1991] Wiederhold, Jajodia and Litwin. Dealing with Granularity of Time in Temporal Databases. Proc. 3rd Nordic Conf. On Advanced Information Systems Engeneering, May 1991

6.4.2 Bases de Datos Espaciales

[Gould,] Gould M. Infraestructura de datos espaciales. Departamento de Lenguajes y Sistemas Informáticos. Universitat Jaume I. Disponible en *gvsig-desktop.forge.osor.eu/.../Infraestructuras_de_Datos_Espaciales.pdf*

[Güting, 1994] Güting R. An introduction to Spatial Database Systems. VLDB Journal, Oct. 1994.

[MySql] MySQL 5.0 Reference Manual.

[Shekhar and Chawla, 2003] Shekhar S. & Chawla S. Spatial Databases: A Tour. Prentice Hall, 2003. ISBN 013-017480-7. Disponible en www.spatial.cs.umn.edu/Book/. 2003.

[Yeung et al., 2007] Yeung A., Hall K.W., Brent G. Spatial Database Systems. Design, Implementation and Project Management Series: GeoJournal Library, Vol. 87 2007, XII, 554 p. ISBN: 978-1-4020-5391-7. 2007.

6.4.3 Bases de Datos Espacio Temporales

[Chen et al, 2001] Chen C., Cardenas A. F., Stott Parker D. and Valentino D. Data Models and Query Languages of Spatio-Temporal Information, 2001.

[Chen et al, 2000] Chen C., Zaniolo C. SQLST: A Spatio-Temporal Data Model and Query Language. En Proceedings of the 19th International Conference on Conceptual Modeling, (ER'00) , pp 96-111. 2000.

[Li and revesz, 2003] Li L. & Revesz P. The relationship among GIS-oriented spatiotemporal databases. Proceedings of the 2003 annual national conference on Digital government research. 2003.

[Munout and Zimanyi, 2004] Minout M. & Zimanyi E. Algebra to SQL query translation for spatiotemporal databases. LNCS Springer ISSN 0302-9743 (Print) Volume 3180 / 2004.

[Parent et al., 1999] Parent C., Spaccapietra S. & Zimányi E. Spatiotemporal conceptual models: data structure + space + time. In C. Medeiros, ACM GIS (pp.26-33). ACM Press. 1999

7 — Sistemas de Bases de datos NoSQL

Elaborado por : Msc María Hallo
Escuela Politécnica Nacional
Quito Ecuador

7.1 Introducción

El término NoSQL significa not only SQL. Se usa para agrupar sistemas de administración de bases de datos diferentes a los tradicionales relacionales. Los conceptos de *Big Data, Big Users, and Cloud Computing* han motivado fuertemente el desarrollo de la tecnología NoSQL. Google, Amazon, Facebook, Mozilla, Adobe, McGraw-Hill Education, Linkedln son algunas de las companías que descubrieron las limitaciones de la tecnología relacional y desarrollaron sus propios sistemas. Algunos proyectos *open source y* otros comerciales se han desarrollado para apoyar estas iniciativas. Amazon desarrolló DynamoDB (Big data and NoSQL), Google usa Big table y Facebook crea y hace uso de Cassandra.

El término NoSQL fue usado en 1991 en un artículo de Unix y posteriormente en 1998 se lo utilizó para nombrar a una base de datos open-source, la misma que no tenía una interface SQL aunque si seguia el modelo relacional, ver caracteristicas del sistema en: `http://www.linuxjournal.com/article/3294`.

En el año 2009, Eric Evans reutilizó el término para un evento sobre bases de datos no relacionales, distribuidas y no conformes a las propiedades ACID propias de los sistemas relacionales. A partir de ese año se ha observado un incremento significativo de las tecnologías NoSQL.

Los sistemas tradicionales relacionales implementaron en un solo sistema un gran número de características tales como modelo de datos relacional, lenguajes de consulta declarativos, control de transacciones, eficiencia, concurrencia, seguridad y almacenamiento persistente de grandes cantidades de datos muchas de ellas no necesarias para determinados sistemas.

Los nuevos requerimientos en la época actual incluyen: disponibilidad total, tolerancia a fallos, almacenamiento de penta bytes de información distribuida en miles de servidores, buen desempeño sin límite de nodos con escalabilidad horizontal, manejo de grandes cantidades de datos estructurados y no estructurados, nuevos métodos de consulta, libertad para manejar nuevos esquemas y datos con control total sobre ellos.

Como ejemplos Facebook añade diariamente más de 15 TB de datos provenientes de registros de clicks y maneja alrededor de 135 billones de mensajes por mes (Lars, 2011).

Los sistemas de Bases de Datos NoSQL se han enfocado en estos nuevos requerimientos y se consideran las bases de datos de la edad del Internet. Nacen de la necesidad de manejar grandes cantidades de información en aplicaciones que soportan millones de usuarios por día, capturando por Internet información personal, de redes sociales, de contenidos generados por el usuario, de información geo localizada entre otros. Estos tipos de sistemas de manejo de datos no requieren

esquemas fijos, son fáciles y rápidos en la instalación, usan lenguajes no declarativos, ofrecen alto rendimiento y disponibilidad, evitan operaciones de junturas, soportan paralelismo y escalan principalmente en forma horizontal soportando estructuras distribuidas que no necesariamente tienen las propiedades *ACID*, *(atomicity, concistency, isolation, durability).*

7.2 Sistemas de Administración de bases de datos NoSQL

Existen diferentes tipos de Sistemas de Administración de Bases de Datos NoSQL, los mismos que manejan varios tipos de bases de datos.

Los sistemas NoSQL se pueden agrupar en:

1. Almacenamientos clave-valor, para requerimientos similares a los *OLTP (online transaction processing)* con muchas transacciones sobre pequeñas cantidades de datos, incluyen los sistemas basados en columnas.
2. Almacenamientos de documentos
3. Bases de Datos de grafos

7.2.1 Almacenamientos clave-valor

El modelo de datos está formado de pares clave-valor.

Para la implementación se usa una tabla hash con una clave y un puntero al objeto correspondiente. Es simple y de fácil implementación. El sistema provee tolerancia a fallos y escalabilidad.

La eficiencia se implementa distribuyendo los registros en base a valores claves, la tolerancia a fallos mediante replicación, las transacciones incluyen registros simples lo cual evita la implementación de protocolos como el de dos fases.

Ejemplos de sistemas clave-valor son: Google BigTable, Amazon Dynamo, Cassandra, Voldemort, HBase.

7.2.1.1 Cassandra

Cassandra es un Sistema de Administración de Bases de Datos clave-valor pero con un modelo de datos de la familia de columnas. Fue liberado como open-source por facebook en Julio del 2008 y escrito en java con influencia de diseño de Dynamo y del modelo de datos de Google Bigtable. Cassandra fue aceptado en el grupo Incubator en 2009 y es usado en producción por Facebook, Twitter, Cisco, Digg entre otros.

Ofrece un modelo de datos libre de esquemas y puede almacenar cientos de terabytes de datos (Hewitt et ál, 2011).

A diferencia de los sistemas relacionales, Cassandra no implementa integridad referencial, no tiene un lenguaje de consultas ni índices secundarios. Por otra parte implementa funciones MapReduce.

Las implementaciones MapReduces aparecen en varios sistemas noSQL Inician con Google junto con el sistema de archivos distribuido HDFS (Hadoop Distributed File System). Ofrecen al usuario acceso mediante funciones específicas tales como map(), reduce(), reader(),writer().

La función map divide el problema en sub problemas (0 o más pares clave–valor)

La función reduce trabaja en los sub problemas y combina los resultados produciendo 0 o más registros a partir de los parámetros clave-lista_de_valores.

El diseño de Cassandra está orientado a optimizar el rendimiento al trabajar sobre muchas máquinas distribuidas aun en lugares geográficos dispersos. Todos los nodos trabajan de la misma manera, con un diseño de centralizado implementando una simetría de servidores. En esta estructura no hay un solo punto de fallos por lo que si un nodo falla los otros continúan trabajando con lo cual existe gran disponibilidad de los servicios. Además se pueden replicar los

datos en múltiples centros de datos para ofrecer mejor rendimiento local y prevenir cortes de servicios en caso de catástrofes que afecten a algún nodo.

Cassandra ofrece un nivel de consistencia configurable (consistencia eventual), lo cual significa que todas las actualizaciones se propagan en todas las réplicas en un sistema distribuido aunque esto puede tomar algún tiempo, pero no se bloquean las actualizaciones en caso de fallos temporales de uno o varios nodos. Este diseño está basado en el teorema de Brewer que señala que de las propiedades: consistencia, disponibilidad y tolerancia a particionamiento solamente dos de las tres se pueden alcanzar al mismo tiempo lo que significa que se deben hacer concesiones sobre ellas.

Con respecto al modelo de datos en Cassandra se almacenan pares nombre /valor que constituyen las columnas. Se pueden definir además familias de columnas para asociar datos similares. Ej se pueden tener familias de columnas de usuarios, ciudades etc en forma similar a una tabla en un modelo relacional pero con un número variable de columnas. Cada columna tiene una marca de tiempo que señala la última fecha en que fue modificada la columna.

Ej en notación JSON (Java script Object Notation):

Ciudades (Familia de Columnas)
Quito: Clavefila
población: 2.000.000 (Nombre de columna:valor)
alcalde: Agusto Barrera(Nombre de columna:valor)
Cuenca: Clave-fila
población: 400.000(Nombre de columna:valor)
alcalde: Paul Granda(Nombre de columna:valor)
exporta: artesanias(Nombre de columna:valor)

Cassandra no garantiza aislamiento en ejecución de transacciones distribuidas, no tiene control de transacciones por lo que si una operación de inserción falla se debe deshacer manualmente las transacciones incompletas.

Una fila en una familia de columnas contiene pares: nombre de columna-valor no necesariamente en igual número en todos los casos. Cada fila tiene una clave.

Las consultas son definidas con los términos *range* y *slice.Range* permite definir un rango de claves para recuperación de filas. El término *Slice* es usado para referirse a un rango de columnas en una fila. Las operaciones de inserción y consultas se realizan usando un API proporcionado por Cassandra. Existe una variedad de interfaces cliente disponibles para varios lenguajes, se puede ver una variedad de opciones en el wiki: `http://wiki.apache.org/cassandra/ClientOptions`. Por otra parte se tiene también herramientas de monitoreo y mantenimiento del cluster de Cassandra para obtener estadísticas, balanceo de nodos, respaldos, etc. Existe además mecanismos de integración con Hadoop, conjunto de proyectos open source para manejo de grandes cantidades de datos en ambientes distribuidos con herramientas analíticas tales como Pig and Hive que posibilitan el uso de funciones de mapeo y reducción.

7.2.1.2 HBASE

HBase es un sistema de bases de datos distribuido que soporta almacenamiento estructurado de datos para tablas grandes.

Hbase fue creado en el 2007 por Powerset para luego convertirse en un proyecto de la Fundación de software Apache. Diferentes versiones fueron liberadas desde entonces. En *Mayo* de 2010 HBase se convierte en el proyecto principal de Apache

Hbase no tiene un lenguaje de consulta declarativo como parte de la implementación principal y tiene limitado soporte para transacciones.

Hbase se usa de preferencia con Linux o cualquier sistema similar que soporte java. Ej. Fedora, CentOS, Debian,.Solaris, Red Hat Enterprise Linux. HBase no se ha probado sobre Windows por lo que no se recomienda utilizar ese sistema operativo.

Hbase tiene las siguientes características, es distribuido, escalable, permite almacenar tablas muy grandes de información en el órden de billones de tuplas y millones de columnas con totales de terabytes o pentabytes de información distribuida en miles de máquinas. El formato de almacenamiento usado es ideal para leer pares clave valor y es optimizada para operaciones de Entrada/Salida de bloques (Lars,2011).

7.2.2 Almacenamientos de documentos

El modelo de datos está compuesto de tuplas (clave, documento).Es similar al almacenamiento clave-valor pero requiere que los datos del objeto documento se encuentren en un formato específico tal como JSON, XML y otros formatos semi estructurados. Los objetos documento son usualmente otras colecciones clave-valor con valores anidados asociados con cada clave.

Las principales operaciones que pueden realizarse sobre la base de datos son: *Insert(key-document), fetch(key), update(key), delete(key), fetch* (búsqueda) basado en el contenido del documento.

Ejemplos de sistemas de manejo de documentos son: CouchDB, MongoDB, SimpleDB.

7.2.2.1 COUCHE BD

Es un sistema de Administración de Bases de Datos orientado a documentos liberado bajo la licencia de código abierto de Apache. Su desarrollo se inicia en el 2005 basado en principios de Lotus Notes y se libera la primera versión en el 2009. Según el wiki de CouchDB el nombre proviene de "*Cluster Of Unreliable Commodity Hardware*", lo cual indica que el software corre en forma distribuida en un cluster de servers económicos. Sus autores predicen que escalará bien sin pérdida de confiabilidad y disponibilidad(Lenon,2009). En CouchDB los datos se almacenan en una serie de documentos, cada uno de ellos con sus campos y se ofrece un modelo basado en java script para realizar agregaciones y reportes sobre los datos. Los documentos son independientes, cada uno de ellos con sus propios campos. Los valores a almacenarse pueden ser cadenas de caracteres, números, fechas, valores booleanos, listas, mapas u otros tipos de datos y metadatos.El sistema no ofrece control de concurrencia de manera que si dos usuarios trabajan sobre un mismo documento, luego de la grabación de uno de ellos, el segundo perderá sus cambios. Por otra parte CouchDB no sobre escribe documentos existentes, más bien guarda una nueva copia. Las copias anteriores desaparecen cuando se compacta la base de datos. Desde el 2008 tiene soporte para las funciones Map/Reduce lo cual permite el manejo eficiente de consultas sobre grandes cantidades de datos particionadas. Por otra parte CoucheDB permite también realizar funciones de agregación, replicación, compactación y operaciones similares a junturas. Para desarrollo de aplicaciones se cuenta con librerías en Python y Ruby.CouchDB implementa las propiedades ACID (*Atomicity, Concurrency, Isolation, Durability*) de una transacción.

A continuación se indican algunos comandos curl para realizar operaciones sobre las bases de datos.

La creación de una base de datos de libros: Se realiza utilizando el comando:
$ curl -X PUT `http://......./libros`.

Para crear el documento libro1:
$ curl -X PUT http://......... /libros/libro1 -d '{}'El comando anterior crea el documento libro1 y asigna un ID (número de revisión).

Para recuperar el documento:

$ curl -X GET http://........ /libros/libro1.

Para borrar el documento libro1:
$ curl -X DELETE http://....../libros/libro1?rev=...........

Para consultar los documentos en libros:
$ curl -X GET http://....... /libros/_all_docs?descending=true

CoucheDB utiliza Futon basada en web, como herramienta de administración para realizar tareas de creación y administración de bases de datos y documentos.

7.2.2.2 MongoDB

MongoDB (de la palabra en inglés “humongous” que significa enorme) es un sistema de base de datos NoSQL libre y de código abierto, orientado a documentos. MongoDB fué liberado en el 2009, trabaja sobre una base de datos de documentos JSON *(JavaScript Object Notation)*, los datos son almacenados en una forma binaria de JSON conocida como BSON. Un documento puede encadenarse a una fila de una tabla relacional con valores que pueden anidarse a una profundidad arbitraria (Zachari et al.,2013).MongoDB no soporta junturas del lado del servidor. Para acceso al documento y su contenido anidado se puede usar llamadas JavaScript.

Dentro de MongoDb se pueden almacenar un conjunto de bases de datos. Una base de datos contiene un conjunto de colecciones las cuales no reforzan esquemas predefinidos. Una colección contiene documentos. Los documentos en una colección tienen un conjunto de campos no necesariamente los mismos para todos.Cada campo es un par clave valor. Una clave es un nombre de campo.Un valor puede ser un entero, flotante, una cadena de caracteres, un documento o un arreglo de valores.

Cualquier campo en un documento en MongoDB puede ser indexado. Se pueden anidar estos valores a cualquier profundidad.

Ejemplos de operaciones sobre MongoDB:

Creación de una colección ciudad.

> db.ciudad.insert({

nombre: *"Quito"*,

población: 1600000

famosa_por: [*"museo mitad del mundo"*, .*escultores"*],

alcalde: {

nombre: .*Agusto Barrera"*,

partido : .*AP"*

}

}

Consulta de la colección ciudad:

> db.ciudad.find()

{

"_id": ObjectId(″. . . . .″),

"nombre": "Quito",

"población": 2‘239.190,

"famosa_por": [″museo mitad del mundo", .escultores″],

.ªlcalde": { "nombre": .Augusto Barrera″, "partido": .AP"}

}

Los símbolos {...} representan un objeto y [...] representan un arreglo.

El campo Objectid es de 12 bytes compuesto de una marca de tiempo, el ID de la máquina del cliente, el ID del proceso del cliente y un contador de 3 bytes.

En MongoDb se pueden construir consultas por rangos de valores de campos o combinaciones de criterios.

Para la consulta de ciudades que tienen una población menor.que 200.000 habitantes se puede usar la expresión:

db.ciudad.find(

{ población : { $lt : 10000 } },

{ nombre : 1, población : 1 }

){

"nombre": *″Puyo″*, "población":100.000 }

Mongo es usado actualmente como back end en varios sitios web tales como eBay, NewYork Times, siendo una de las bases de datos NOSQL más populares.

7.2.3 Sistema de Base de Datos de Grafos

El modelo de datos comprende en las bases de datos de grafos está formado de nodos y aristas.

Los nodos pueden tener propiedades y las aristas pueden tener etiquetas y roles.

Los datos de grafos se almacenan en tuplas con múltiples atributos. Se pueden representan

como grafos datos de mapas de rutas, redes sociales, datos de recursos y sus relaciones.

Ejemplos de sistemas de bases de datos de grafos son : Neo4j, FlockDB, Pregel.

Por otra parte los triples RDF pueden ser mapeados a bases de datos de grafos.

7.2.3.1 Neo4j

Neo4j es un tipo de sistema de administración de bases de datos de grafos. Esta herramienta permite diseñar grafos gráficamente. Permite almacenar billones de nodos y aristas. Los nodos y aristas pueden contener propiedades y valores.Neo4j es más enfocada a almacenar relaciones entre valores lo cual permite almacenar gran cantidad de datos-

Hay varios lenguajes que interoperan con Neo4j: Cypher, Consola de Ruby, Gremlin, interface REST, los cuales permiten realizar consultas tales como los nodos que cumplen con una condición, los nodos relacionados con una arista, aristas conectados con un nodo, el camino de acceso entre dos nodos (path) y aplicar funciones de mapeo y reducción. Para acelerar las búsquedas Neo4j permite construir índices por valores de claves e incorpora Lucene para construir índices invertidos.

Neo4j es un sistema de administración de bases de datos que garantiza las propiedades de aislamiento, consistencia, atomicidad y permanencia de una transacción (Redmond et ál., 2012).

7.3 Tecnologías de Linked Data y repositorios RDF

Linked Data es un término propuesto por Tim Berners-Lee (2006) y es usado para describir un conjunto de prácticas recomendadas para publicar, compartir y conectar piezas de datos, información y conocimiento en la Web semántica usando identificadores URIs (Uniform Resource Identifiers) y RDF (*Resource Description Famework*) para describir los recursos (Hartig et ál., 2010). RDF usa para los datos modelos de grafos.

Se pueden distinguir varios períodos en la evolución de *Linked Data*. Entre los años 1990 y 2000 se realizaron varios desarrollos de software para introducir representación de conocimiento principalmente en la Web de documentos. En 1999 se liberó la primera versión de RDF. La idea fue desarrollar una capa RDF sobre XML con el fin de poder usar las tecnologías como XML y XSLT. Entre 2000 y 2006 se estabilizó RDF, se añadió una semántica formal y se liberó OWL (Hausenblas, 2009). En este período existe escepticismo en que se pueda llegar a compartir datos, lo cual es actualmente superado en varios gobiernos gracias al movimiento *LinkedOpen Data* que impulsa el fácil acceso de ciudadanos y empresas a los datos públicos que recogen lasAdministraciones Públicas.

En el año 2006, Tim Berners_Lee escribió una nota de diseño (Berners-Lee, 2006), proponiendo soluciones a los problemas que impidieron el enlace de los datos, mediante la aplicación de los principios de *Linked Data* que son los siguientes:

- Use URIs como nombres de recursos
- Use HTTP URIs de manera que la gente pueda buscar esos nombres
- Cuando alguien busca un URI deberían encontrar información útil usando los estándares RDF o SPARQL
- Incluya enlaces a otros URIs de manera que la gente pueda encontrar más recursos relacionados.

Lo anterior implica usar el modelo de datos RDF para publicar datos estructurados en la web y usar enlaces RDF para enlazar los datos de diferentes fuentes. Estos principios permitieron impulsar el desarrollo de varios proyectos de *Linked Data* (Heath et ál., 2011).

A partir del 2007 se ha producido un avance más rápido en el desarrollo de tecnologías de *Linked Data*. Se han liberado algunos estándares de la W3C tales como: SPARQL, GRDDL, RDFa, se ha formado la comunidad del proyecto *Linking Open Data* y cada vez se observa un

creciente uso en gobierno electrónico con numerosos catálogos de datos publicados (Berners-Lee, 2009; Sheridan et ál., 2010, Cyrille et ál., 2010).

7.3.1 Sistemas de gestión de repositorios RDF

Existen varios sistemas de gestión de repositorios RDF desarrollados en los últimos años. Entre ellos se pueden citar: Virtuoso Universal Server, Apache JENA, Sesame, Sparkle DB, Ontotext_OWLIM, 3store, Mulgara.

Los sistemas indicados ofrecen almacenamiento persistente y acceso a grafos RDF mediante el lenguaje de consulta SPARQL lo cual puede ser construido como parte de la herramienta principal (Sesame) o en forma separada (Jena TDB). Algunos permiten razonamiento con OWL pero no todos. Citaremos a continuación características de dos de ellos.

7.3.1.1 Virtuoso Universal Server

Combina en un solo sistema la funcionalidad de los sistemas RDBMS(Relational Database Management System), ORDBMS(Object Relational Database Management System), RDF, XML, servidor de aplicaciones web. La edición de código abierto de Virtuoso se conoce como OpenLink Virtuoso.

Para las consultas RDF se utiliza Sparql, para XML, XQuery y SQL para los datos relacionales. Permite control de transacciones y recuperación en caso de fallos con el manejo de registros de transacciones y bloqueos. Garantiza las propiedades ACID (atomicity, consistency, isolation, durability) de una transacción. El sistema permite además generar datos enlazados en RDF a partir de fuentes de datos relacionales. Para obtener una mejor explotación de este sistema hace falta ampliar los tutoriales y documentación existente con ejemplos de implementación de casos de estudio.

7.3.1.2 Apache Jena TDB

Es un componente del marco de trabajo Jena, usado para construir aplicaciones de web semántica. TDB maneja el almacenamiento y consulta de grafos RDF usando SPARQL y SPARUL, soporta las propiedades ACID de una transacción y permite accesos mediante el API de Jena para el desarrollo de aplicaciones.

Jena soporta serialización de grafos a bases de datos relacionales, RDF/XML, TURTLE entre otros.

Para el acceso se puede usar Fuseki, una interface que usa Sparql para actualizaciones y consultas.

7.3.2 Tecnologías relacionadas con *Linked Data que interactuan con repositorios RDF.*

Varias tecnologías están ya disponibles para publicación y consumo de *Linked Data*. En este apartado se clasifican atendiendo a su finalidad: técnicas y herramientas que sirvan para la publicación, o técnicas y herramientas destinadas a facilitar el consumo de datos enlazados (Hallo et ál (2012). Los datos reposan en respositorios RDF (sujeto,predicado,objeto).

7.3.2.1 Publicación

La publicación en la Web semántica se puede realizar usando una variedad de herramientas y patrones de publicación siguiendo los principios de *Linked Data,* lo cual permite la interoperabilidad de los datos y reutilización en la Web. Sin embargo esto no implica el abandono de los sistemas existentes si no más bien la adición de capas extras para conectar los datos en la Web. Estas capas pueden estar formadas 'por extractores de entidades para armar archivos RDF, almacenes de datos RDF, interfaces a sistemas existentes, uso de Sistemas de Administración de Contenidos (CMS) con salidas RDF entre otros (García et ál., 2011).

Para publicar *Linked Data* se han desarrollado varias herramientas entre las cuales tenemos:

- *D2R server* es una herramienta para publicar bases de datos relacionales como *Linked Data* transformando el contenido de una base de datos relacional a RDF (Cyganiak et ál., 2009).
- *Triplyfy* es un plugin para aplicaciones web que revela las estructuras semánticas codificadas en bases de datos relacionales haciendo el contenido de bases de datos disponibles como RDF, JSON o *Linked Data* (Jaenicke, 2009).
- La Plataforma *Talis* proporciona hospedaje para contenido y datos RDF tanto para publicar datos como para desarrollo. Además ofrece interfaces de consulta y manipulación de datos (Bizer et ál., 2008).
- *Pubby* es una interface para Linked Data que permite interactuar con puntos SPARQL (SPARQL endpoints) mediante consultas SPARQL (Cyganiak et ál., 2011).
- *Open Link Virtuoso Universal Server*: Es un sistema de administración de bases de datos orientado a objetos que permite almacenar datos RDF y presentarlos con la funcionalidad de un punto SPARQL. Los datos en RDF pueden ser almacenados directamente en Virtuoso o creados a partir de bases de datos relacionales no RDF (Erling, 2010).
- *Virtuoso Sponger*: Permite transformar datos no RDF en RDF. Acepta como entradas páginas web (X)HTML, páginas web (X)HTML con microformatos y aun servicios web como aquellos de Google, Del.icio.us, Flickr etc. Y crea RDF como salidas (Bizer et ál,2007).
- *Sesame*: Es un sistema formado por un repositorio, una máquina de búsqueda y un módulo de administración para añadir y borrar datos RDF e información de esquema. *Sesame* está siendo usado en algunas grandes compañias y agencias de gobierno para integración de datos (Broekstra et ál, 2004).
- *JenaTDB*: Es un repositorio de datos RDF que permite añadir datos usando SPARQL/Update y permite consultas de datos usando el lenguaje SPARQL (Heath, T. et ál, 2011).
- *3Store* : Es un repositorio de triples basado en MySQL. El software no presenta interfaces directamente al usuario pero puede ser consultado por varios servicios incluyendo el navegador Direct RDF (Aduna,2009).

Editores de Linked Data y Validadores.

- Hyena: Editor RDF, una mezcla de una wiki y una base de datos, disponible como aplicación de escritorio y como aplicación web (Rauschmayer, 2010).
- Vapour: Validador de Linked Data (García et ál., 2011).
- DriftR : Editor y navegador de Linked Data (Nowack, 2007).

Existen otras herramientas, pero aquí se han seleccionado las que se han considerado más representativas, bien por su estado de avance, o por su grado de utilización de las tecnologías asociadas a *Linked Data*. Un listado, que se viene actualizando con regularidad, se puede encontrar en la wiki sobre herramientas de Web semántica, publicada y mantenida por el consorcio W3C (W3C, 2010).

Para facilitar la publicación de datos enlazados se puede migrarlos a RDF desde bases de datos relacionales, hojas electrónicas, XML, sitios web existentes tras ser indexados y puestos a disposición para consultas vía un servicio SPARQL. Los datos de salida pueden obtenerse en formatos como XML, RDF. Cada conjunto de datos debe tener sus metadatos y se los debe catalogar para facilitar las búsquedas (Berners-Lee, 2009).

Como ejemplo se puede citar el proyecto *DBpedia* que usa scripts de PHP para extraer datos

estructurados desde páginas de Wikipedia. Estos datos son convertidos a RDF y almacenados en el repositorio *OpenLink Virtuoso* el mismo que proporciona un punto de acceso SPARQL.. Para acceder a los datos se usa *Pubby* (Heath,2011).

7.3.2.2 Consumo de *Linked Data*

Los datos publicados en la Web de acuerdo a los principios de *Linked Data* forman parte de un espacio global de datos. *En general en estos casos las aplicaciones explotan las propiedades de Linked Data entre las que podemos citar (Bizer et ál., 2007):*

a) Representación y acceso estandarizado de datos que poseen descripciones.
b) *Arquitectura abierta que permite descubrir nuevos datos en tiempo de ejecución a medida que están disponibles en la Web semántica.*

Las tecnologías de Web semántica soportan un flujo de información punto a punto. Para requerimientos de integración, consultas e inferencias son útiles RDF, RDFs, OWL, SPARQL (Harris et ál., 2008).

Para acceso a *Linked Data* se han desarrollado varias herramientas:

Se puede usar navegadores específicos (*Linked Data browsers),* permitiendo al usuario desplazarse entre diferentes fuentes de datos siguiendo los enlaces *RDF.* Por otra parte los documentos de la Web semántica pueden ser rastreados automáticamente siguiendo los enlaces RDF y los datos así obtenidos pueden ser sometidos a más sofisticadas capacidades de búsqueda (Heath et ál., 2011).

Entre los navegadores de datos RDF tenemos *Object Viewer,* Marbles, Disco, Tabulator, OpenLink RDF Browser, Zitgist Data Viewer, Dipper. Razorbase. iLOD, Fenfire, Quick & Dirty RDF Browser. En cuanto a rastreadores RDF se tiene SWSE, Swoogle, *URIBurner,* los mismos que proporcionan descripciones RDF de recursos web en una variedad de formatos (Hartig et ál., 2010).

Existen máquinas de búsqueda de Linked Data, tales como Sig.ma, *Falcons* y *SWSE* que proporcionan servicios de búsqueda, por palabras clave orientados a los usuarios. Adicionalmente, proporcionan enlaces a los documentos fuente que tienen las palabras clave mencionadas, lo cual ofrece mejores capacidades de interacción para explotar la estructura de los datos. Por ejemplo, se puede filtrar resultados por clase y limitar los resultados a una subclase específica u obtener una lista de resultados con enlaces a entidades relacionadas (Hartig et ál., 2010).

Por otra parte se han desarrollado servicios sobre datos enlazados distribuidos como Sindice, que proporciona una API para que las aplicaciones puedan encontrar documentos RDF en la Web que se refieran a una cierta URI o contengan ciertas palabras clave. Estas palabras clave sirven para caracterizar las instancias de determinadas entidades de datos. Esto permite acceder a documentos de la Web semántica que contienen ciertas instancias de datos (Hausenblas, 2009).

También se han desarrollado algunas herramientas para apoyar el consumo y fusión de fuentes de *Linked Data.* Es el caso de ALOE (*Asisted Linked Data Consumption Engine*) (AxelNgonga, 2011), que permite obtener información proveniente de varios servidores en la web como triples N3. Se usa para los ejemplos el repositorio *DR2 Server Publishing* (Cyganiak et ál.,2011).

7.3.3 Respositorios RDF basados en tecnologías NOSQL

Existen en experimentación propuestas de diseño de repositorios RDF usando tecnologías NOSLQ. Torre, A, et ál(2011) sugieren almacenar las tripletas RDF(sujeto, predicado, objeto) en sistemas de bases de datos orientados a columnas como Cassandra debido a sus propiedades de escalabilidad y rendimiento y a su naturaleza distribuida. Para las consultas se sugiere usar SPARQL mediante un módulo intermedio que permita realizar la traducción desde este lenguaje

a la consulta en bases de datos NoSQL asi como usando indexación distribuida basada en MapReduce, tecnología proveniente del mundo NOSQL.

Actividades para el estudiante:

- Realizar un estudio de 10 aplicaciones en ejecución que trabajen con bases de datos NOSQL indicando las ventajas con respecto a sistemas relacionales
- Realizar un estudio de 10 aplicaciones en ejecución que trabajen con repositorios RDF indicando los servicios que ofrecen.

7.4 Bibliografia

Aduna (2009*). OpenRdf.Org,* http://www.openrdf.org/index.jsp (2011-08-5).

Allemang, D. y Hendler, J. (20*11). Semantic Web the working Onto*logist, Waltham, Morgan Kaufmann,2011.

Becker, C. y Bizer C. (2009). Exploring the geospacial semantic web with dbpedia mobile. *//Journal of Web Semantics*, **7**(4) 278-286.

Berners-Lee, T. (2006). *Design Note: Linked Data*. http://www.w3.org/DesignIssues/LinkedData.html (2011-07-02).

Bizer, C.;Heath T., et ál. (2008). Linked data on the web (LDOW2008). Proceeding of the 17th international conference on World Wide Web. ACM. Beijing, China**:** 1265-1266.

Brunozzi,S (2012). Big Data and NoSQL with Amazon DynamoDB. In *Proceedings of the 2012 workshop on Management of big data systems* (MBDS '12). ACM, New York, NY, USA, 41-42.

Cyganiak, R. y Bizer C. (2011). *Pubby.A Linked Data Frontend for SPARQL Endpoints*. http://www4.wiwiss.fu-berlin.de/pubby/ (2011-07-02).

Harris, S.;Gibbons J., et ál. (2008). Semantic technologies in electronic government. Proceedings of the 2nd international conference on Theory and practice of electronic governance. ACM. Cairo, Egypt**:** 45-51.

Hallo,M, De la Fuente, P., Martínez-Gonzáles M.(2012). Las tecnologías de Linked Data y el gobierno electrónico. SCIRE, España,18(1),49-61.

Hartig, O. y Langegger A. (2010). *A Database Perspective on Consuming Linked Data on the Web*. http://www2.informatik.hu-berlin.de/{~}hartig/files/Hartig_QueryingLD_DBSpektrum_Preprint.pdf (2011-07-04).

Hartig, O.;Sequeda J., et ál. (2010). *How to consume Linked Data on the Web*. http://iswc2009.semanticweb.org/wiki/index.php/ISWC_2009_Tutorials/How_to_Consume_Linked_Data_on_the_Web (2011-07-05).

Hausenblas, M. (2009). *Linked Data Applications DERI Technical Report 2009-07-26*. http://

wtlab.um.ac.ir/parameters/wtlab/filemanager/LD_resources/other/lod-app-tr-2009-07-26_0.pdf (2011-06-20).

Heath, T. y Bizer C. (2011). *Linked Data: Evolving the Web into a Global Data Space*, Morgan & Claypool,2011.

Hewitt, E. et al. (2011). Cassandra: The Definitive Guide ,O'Reilly Media, Inc.USA

Jaenicke,N.(2009). *Triplify*, http://triplify.org/Overview (2011-07-06).

Lars, G. (2011), Hbase the definitive guide, O'Reilly Media, Inc. USA,2011.

Lenon, J.(2009), Beginning CouchDB, Apress,Springer-Verlang,2009

Parker,Z. et ál.(2013). Comparing NoSQL MongoDB to an SQL DB. In *Proceedings of the 51st ACM Southeast Conference* (ACMSE '13). ACM, New York, NY, USA, , Article 5 , 6 pages.

Pokorny, J. (2011). NoSQL databases: a step to database scalability in web environment. In *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services* (iiWAS '11). ACM, New York, NY, USA, 278-283.

Rauschmayer, A. (2010). *Hyena: Organize your ideas*. http://proj.2ality.com/hyena/ (2011-07-02).

Redmond,E. et ál.(2012). Seven Databases in Seven Weeks A Guide to Modern Databases and the NoSQL Movement, Dallas, Texas • Raleigh, North Carolina. 2012.

Sattar,A. et ál (2013). Incorporating NoSQL into a database course. *ACM Inroads* 4, 2 (June 2013), 50-53.

Sheridan, J. (2010). *Legislation.gov.uk*. http://blog.law.cornell.edu/voxpop/tag/legal-linked-data/ (2011-05-16).

Schindler, J. (2012). I/O characteristics of NoSQL databases. *Proc. VLDB Endow.* 5, 12 (August 2012), 2020-2021.

Torre-Bastida, A. Bermúdez,J. Illarramendi A. and González M.(2011). *Diseño de un repositorio RDF basado en tecnologías NOSQL*. 16th Conference on Software Engineering and Databases (JISBD 2011), A Coruña(Spain), Sep. 2011

Proyecto LATIn

Edición: Marzo de 2014.
Este texto forma parte de la Iniciativa Latinoamericana de Libros de Texto abiertos (LATIn), proyecto financiado por la Unión Europea en el marco de su Programa ALFA III EuropeAid.